1991

An Adaptive Differencing Scheme for Elliptic Flows.

Therese Estelle Rhodes
Louisiana State University and Agricultural & Mechanical College

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

U·M·I

University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600

Order Number 9207526

An adaptive differencing scheme for elliptic flows

Rhodes, Therese Estelle, Ph.D.

The Louisiana State University and Agricultural and Mechanical Col., 1991

U·M·I

300 N. Zeeb Rd.
Ann Arbor, MI 48106

AN ADAPTIVE DIFFERENCING SCHEME FOR ELLIPTIC FLOWS

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Mechanical Engineering

by
Therese E. Rhodes
B.S.M.E., The University of Alabama, 1981
M.S.E.M., The University of Alabama, 1983
August, 1991

ACKNOWLEDGEMENT

I would like to thank my advisor, Dr. Sumanta Acharya, for suggesting the research presented in this dissertation.

This research was conducted using the Cornell National Supercomputer Facility, a resource of the Cornell Theory Center, which is funded in part by the National Science Foundation, New York State, the IBM Corporation and members of the Center's Corporate Research Institute. I thank the staff of the CNSF for their help and advice during the course of this research.

I also thank my teachers and friends for their encouragement and help in acquiring this degree.

Finally, I extend my special thanks and appreciation to my parents. Without their love, support, encouragement and advice I could never have completed this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

This thesis deals with the formulation of a computationally efficient multiple-grid adaptive differencing (MAD) scheme for two-dimensional elliptic flow and heat transfer problems. This algorithm equidistributes a measure of the error by using higher order differencing schemes locally in adaptively determined high error-estimate regions. The third-order accurate QUICK scheme is used in regions of high error estimate which are dynamically flagged on the basis of a preliminary first order upwind solution. Boundary conditions for the flagged regions are taken from the preliminary upwind soultion. Multigrid type calculations are performed.

Three multiple-grid schemes are developed. In the first scheme, MAD1-WFDS, the entire domain and flagged subdomains are solved at each multiple-grid iteration. The second scheme, MAD2-WDS, solves the entire domain at each iteration, employing the QUICK form of the discretized equations in the flagged regions and the original upwind formulation elsewhere. The third algorithm , MAD3-FDS, is similar to the first, except the entire domain is not solved after the subdomain solution. Instead, only the unflagged portion of the problem domain is solved, using the improved values obtained in the flagged regions as boundary conditions.

The three MAD algorithms are applied to two convection-diffusion and two flow problems. The results are compared to the exact solution (if available), the upwind and QUICK

solutions, and to each other. MAD1-WFDS shows the best improvement to the upwind scheme but requires the most additional computing time. MAD2-WDS requires the least additional computing time, but shows the least improvement over the upwind solution.

The code for MAD1-WFDS is parallelized to reduce the real computation time required for problem solutions. The upwind and QUICK schemes are also parallelized for comparison. Several program levels or granularities were parallelized to determine an optimal level of parallelization. Parallelizing on the subdomain level and parallelizing the solution of the general variable equation yielded good results. A real time savings of 26.4% was achieved in one case (in spite of the fact that the solution was not computed on a dedicated machine) at a cost of a 7.8% increase in cpu time required by the parallel run over the serial run.

CHAPTER ONE

INTRODUCTION

1.1 MOTIVATION

With decreasing computer costs and increasing computer power, numerical methods for modeling fluid flow and heat transfer phenomena are becoming more prevalent and sophisticated. In many, if not most instances, the computer model is more cost efficient and less time consuming than the construction of a scale model of a given physical application. However, numerical methods provide information only at a predefined number of discrete points within the problem's domain. In obtaining a numerical solution, the two issues of primary importance are accuracy and economy. Generally, these are competing issues, and an optimal numerical procedure should represent a suitable compromise. The main objective of this thesis is to develop an accurate numerical scheme that minimizes computational effort.

In order to minimize solution error, the conventional procedure has been to optimally redistribute the grid points. It the grid points are clustered in regions of large derivatives of the dependent variable (ϕ), rather than being distributed uniformly, then the solution error, proportional to $h^n f(\phi, \phi', \ldots, \phi^m)$, is equidistributed. However, the critical regions where the derivatives are large are generally not known a priori; hence, a desirable feature of a numerical

code is the ability to identify these critical regions and to redistribute the grid points accordingly. Such a procedure is commonly referred to as an adaptive grid procedure. A second approach, and one that has received practically no attention in the finite difference literature (Kim and Thompson, 1990), is to use higher order differencing schemes in critical regions. Such an approach, which will be called an adaptive-differencing approach in this work, is the focus of the present study.

1.2 TRADITIONAL FINITE DIFFERENCE ADAPTIVE GRID TECHNIQUES

In the finite element literature, and particularly in the solid mechanics area, a classification scheme for the various adaptation strategies has been developed. Methods which redistribute a fixed number of nodes to increase grid density in regions of high error estimate are called r-methods. Schemes which refine mesh sizes (adding more nodes) are referred to as h-methods, while those which involve improving the spectral order of the function approximations on a fixed mesh are classified as p-methods. In the area of finite difference methods, as noted above, current research is concentrated in the area of dynamically adaptive grid systems (h- or r-methods) rather than adaptive differencing techniques (p-method). Existing mesh adaptation techniques in the computational fluid dynamics (CFD) area can be divided into two categories, global and local.

1.2.1 Global Mesh Adaptation Schemes

Global mesh schemes readjust grid point positions to cluster in locations of large gradients and higher order derivatives without employing additional grid points, similar in concept to the r-methods. Typical examples of global mesh refinement techniques include those of Dwyer et al (1982), Gnoffo (1982) and Acharya and coworkers (1982, 1990, 1991, 1991a, 1991b). These methods generally employ a equidistributional type approach, which involves solving a set of algebraic equations or elliptic partial differential equations for each grid generated. The resulting mesh has to be smoothed to minimize errors arising from the grid skewness.

Ushimary (1982) employs the results of a solution on a prelimnary grid to reparameterize the boundary point distribution. His technique rearranges the grid based on simple functions defined for certain regions prior to the solution and therefore is not truly dynamically adaptive. Thompson, Warsi and Mastin (1982) transform the time derivatives in the physical plane to derivatives taken at fixed points in the transformed plane so that when the adpative grid moves, no interpolation is required. This technique is applicable for problems whose grid evolves with the solution at each time step (Thompson, 1982). Gnoffo (1982) developed a similar method but changes the grid at every nth time step.

Variational approaches involving the solution of a set of elliptic partial differential equations for each grid generated have also been used. These techniques minimize a function which is a measure of the grid smoothness, orthogonality and truncation error. Brackbill (1982) and Saltzman et al (1982) explain and develop this approach.

Another group of methods are based on the attraction/repulsion approach developed by Anderson (1982), Rai et al (1981) and Nietubicz (1982). In these methods, an attraction number (based on the difference between the magnitude of some measure of error, say E, and the average magnitude of E_{avg} over all the points) is assigned to each grid point. Then, points with attraction numbers greater than E_{avg} attract neighboring points, while nodes with values less than E_{avg} repel other points. This approach does not require the solution of partial differential equations for each new grid, but neither does it employ any measures to control the grid smoothness and orthogonality. While distortion is possible, the grid points will not collapse into each other since the attraction will become repulsion before that occurs. Some extension and enhancements of the approach have been developed by Greenburg (1983) and Eisemann (1983).

The major disadvantage to the global mesh refinement techniques is the potential loss of accuracy in the "non-critical" regions where the grid point concentration is decreased in order to provide a greater node density in

regions of large gradients. Local mesh methods avoid this problem by adding more grid points in the critical areas.

1.2.2 Local Mesh Refinement Methods

Local mesh refinement techniques add more grid points to critical regions of large error estimate like the h-methods. Phillips and Schmidt (1984) have developed a method for diffusion type problems wherein a fine mesh is used in regions of large gradients and a coarse mesh is preserved in low gradient regions. Solution values at the zonal boundaries where the coarse and fine grids adjoin are updated using interpolation and the different meshes are solved sequentially until convergence. In Berger and Jamenson's (1985) scheme a finer mesh is constructed in the high-error flagged regions by uniformly adding grid points between the existing ones. A new solution is then generated in the flagged region. Caruso et al (1986) developed a scheme employing ideas similar to those of Berger and Jameson (1985). Unfortunately, these methods require the flagged regions to be rectangular in the computational space, thus making them somewhat inefficient.

Braaten and Shyy (1987) have reported a multigrid technique which is a blend of the methods developed by Brandt (1977), Hackbusch and Trottenberg (1982) and Settari and Aziz(1986). Braaten and Shyy (1987) developed their procedure for PISO (see Issa, 1986) and SIMPLE (see Patankar, 1980) type algorithms. The pressure correction equations were solved on

a multilevel grid wherein a 2 x 2 x 2 block of the finest mesh generated is treated as the control volume for the coarser mesh. Then, these coarser control volumes may be considered as the control volume of yet another coarser grid and so forth. In a typical tri-level procedure, the solution is obtained first on the fine grid, followed by the next coarser level and finally the coarsest level. The corrections on the coarsest level are then interpolated for mesh point values on the middle level. Next, the middle level solution is recomputed and its values interpolated onto the fine mesh to complete the cycle. This method may be readily extended to include more intermediate levels as required. This algorithm is shown to improve the convergence rate for laminar flows or flows computed on nearly orthogonal grids, but no significant improvement was demonstrated for turbulent or reacting flows or flows on highly nonorthogonal meshes. Similar approaches have been independently developed by Hutchinson and Raithby (1986), Phillips et al (1985) and Rhie (1986).

A major disadvantage of these methods is the uniform refinement process employed over the entire flagged region (or, in the case of Braaton and Shyy (1987) over the entire domain). With this technique, areas with very high errors and areas with relatively lower errors in the flagged region are refined equally. Hence, computational costs are increased unnecessarily as a result of the inefficient grid distribution.

Caruso (1985), Acharya and Moukalled (1990) and Moukalled and Acharya (1991) used a local adaptive refinement approach which involves overlaying patches of fine grid in regions where the truncation (or solution) error is large. A solution is then calculated on the composite grid. This patching and solving algorithm continues until the desired reduced value of the error estimate is achieved. A recent survey of adaptive grid schemes has been compiled by Eiseman (1987).

1.2.3 Multigrid Techniques

One of the most promising techniques for solving the incompressible Navier-Stokes equations is the multigrid method. These schemes use a sequence of coarser grids to accelerate the numerical model's convergence. Gia et al (1982) first applied this method to the vorticity-stream function formulation of the Navier-Stokes equations. They reported a factor of four decrease in the computational time required as compared with a single grid cacluation. Fuchs and Zhao (1984) applied a multigrid method to the incompressible Navier-Stokes equations in primitive-variable form. Vanka (1986) also developed a multigrid algorithm based on the primitive-variable formulation. His method satisfies continuity for each control volume at each step. He reports a decrease in computational time of a factor of twenty-five over common single grid techniques. De Zeeuw (1990) has achieved an improvement in the multigrid method for nine-point

discretizations of elliptic partial differential equations by automatic adaption of prolongation and restriction operators which are particular to the problem to be solved. Thompson and Ferziger (1989) have coupled Caruso's (1985) automatic adaptive refinement technique with a multigrid approach. Acharya and Moukalled (1990), Moukalled and Acharya (1991) have also developed a multigrid strategy coupled with solution-grid adaptations.

1.3 ADAPTIVE DIFFERENCING TECHNIQUES

As stated earlier, while considerable attention has been devoted to the adaptive grid schemes, adaptive differencing techniques are relatively unexplored in finite difference methods. The idea of employing higher order finite difference schemes to achieve more numerically accurate solutions within a given domain is not, however, a new development. Recently, there has been a great deal of interest in compact fourth-order accurate schemes. Dennis and Hudson (1989) have recently developed such a method which is fourth-order accurate for Navier-Stokes type equations. Rogers and Kwak (1990) employ both third-order and fifth-order differencing schemes for the convective fluxes in the incompressible Navier-Stokes equations. Other methods of this type have been reviewed by Hirsch (1983). Kallinderis and Baron (1989) have developed a scheme which initially employs a coarse grid, over which the full Navier-Stokes equations are solved. Then, a

"feature detection" algorithm is used to adapt (refine) the grid in boundary-layer regions and also to identify appreciably viscous regions. The Navier-Stokes equations are then solved in the viscous regions, while elsewhere, the equations are reduced to the Euler equations. However, in general, the idea of using an adaptive differencing technique has not received attention in the finite difference literature.

Unlike the finite difference field, in the finite element literature a few studies have been reported in adaptive differencing techniques. These p-methods are similar to the spectral element methods of Patera (1986). Demkowicz, et al (1985) first presented an adaptive p-method for two-dimensional Navier Stokes equations. Devloo et al (1988) and Demkowicz et al (1985) developed adaptive p-methods for two-dimensional Navier-Stokes equations. These schemes improve the accuracy of the finite element solution by increasing the degree of the elemental polynomial shape function locally in regions of large error estimate. A recent literature survey of adaptive finite element methods has been assembled by Oden and Demkowicz (1988).

1.4 SCOPE OF THE PRESENT WORK

In this research, an adaptively-differenced, multigrid finite difference technique is developed for the solution of the incompressible Navier-Stokes equations. A higher-order

finite difference scheme is used in dynamically flagged high error estimate regions. As a first step, a preliminary solution is computed over the domain using a conventional, first-order upwind scheme. Next, regions of high error estimates are flagged and a new solution is obtained locally in the flagged region(s) using a higher-order, quadratic upwind scheme (QUICK, Quadratic Upwind Interpolation for Convective Kinematics) developed by Leonard (1979). The conventional upwind solution and the higher order QUICK solution are coupled together in a multigrid manner and calculations continued to the desired accuracy levels.

The algorithm has several levels of calculations and so lends itself to parallel solutions. The different levels of calculations or blocks of code are classified according to coarse and fine granularities. These granularities are explored for possible parallel enhancements to reduce the computation time (real time rather than cpu time) required for problem solutions.

1.5 SURVEY OF THE THESIS

This chapter has surveyed current adaptive grid and adaptive differencing techniques used in solving Navier-Stokes problems. The motivation for this research has been presented.

Chapter Two introduces the solution method for convection-diffusion problems. The governing partial

differential equation is discretized using the first and third order upwind schemes for the convective terms. Two convection-diffusion test case problems are solved using the upwind shceme and the QUICK scheme.

Chapter Three presents three multigrid, adaptive differencing methods. The three methods are applied to the convection-diffusion test case problems of Chapter Two and the results are compared.

Chapter Four extends the solution method to the general flow problem. The multigrid, adaptive differencing methods are then applied to two more test cases. The solution accuracy and computer efforts required for each method are compared.

Chapter Five explores the different levels of parallelization possible for the solution algorithm. Test case results and times are presented.

Finally, Chapter Six is comprised of the major conclusions of this thesis and a discussion of future tasks to be explored.

1.6 CLOSING REMARKS

The objectives of this thesis have been outlined and the motivation for the work explained. The literature survey discusses the currently available work on adaptive grid and adaptive differencing methods in the area of computational fluid dynamics. The next chapter presents the general

convection-diffusion equation with its discretization and solution procedure.

CHAPTER TWO

SOLUTION METHOD FOR CONVECTION-DIFFUSION PROBLEMS

In this chapter, the flow field is assumed to be known. Hence, only the solution for an unknown scalar variable, ϕ, is sought. The governing equation is discretized using two methods for the convection terms, the upwind and QUICK schemes. Next, the general solution method for convection-diffusion problems is presented. Finally, both schemes are applied to two convection-diffusion problems.

2.1 FINITE DIFFERENCE EQUATIONS

The general governing equation for steady convection-diffusion flow may be written as:

$$\frac{\partial}{\partial x}(\rho u\phi)+\frac{\partial}{\partial y}(\rho v\phi)=\frac{\partial}{\partial x}(\Gamma\frac{\partial\phi}{\partial x})+\frac{\partial}{\partial y}(\Gamma\frac{\partial\phi}{\partial y})+S \qquad 2.1$$

where ϕ is the dependent variable, Γ is the diffusion coefficient and S is the source term(s).

2.1.1 Boundary Conditions

The boundary conditions associated with this equation may be classified as inflow, outflow and no flow boundaries. The inflow boundary conditions are defined as the boundary where the flow enters the problem domain. Either the value of the dependent variable ϕ or its first derrivative ($\partial\phi/\partial$x or $\partial\phi/\partial$y) is generally given at the inflow boundary.

Outflow boundary conditions represent flow properties at the exit boundary. Often, these boundary conditions are not known a priori. In the absence of any useful information, the diffusive flux is assumed to be very small and is neglected, i.e., the total flux is assumed to be purely convective. Care must be taken to ensure that the problem domain is sufficiently large to justify this assumption.

Finally, the no flow boundary exists when the flow does not enter or leave the domain. Walls and symmetry lines are typical no flow boundaries. Along a wall, the vlaue of the dependent variable ϕ or its diffusive flux is generally given. At a symmetry line, however, the diffusive flux is assumed to be zero ($\partial\phi/\partial x = 0$).

2.1.2 Domain Discretization

Since the governing equation is discretized using the control volume approach, a discussion of the subdivision of the domain into control volumes is appropriate at this point. Figure 2.1a shows a typical discretized domain for the general variable ϕ. The domain is first subdivided into control volumes and the control volume face locations are decided upon (the grid may or may not be uniform). Next, grid points or nodes are placed at the geometric center of each control volume. The values of the dependent variables are stored at these nodes.

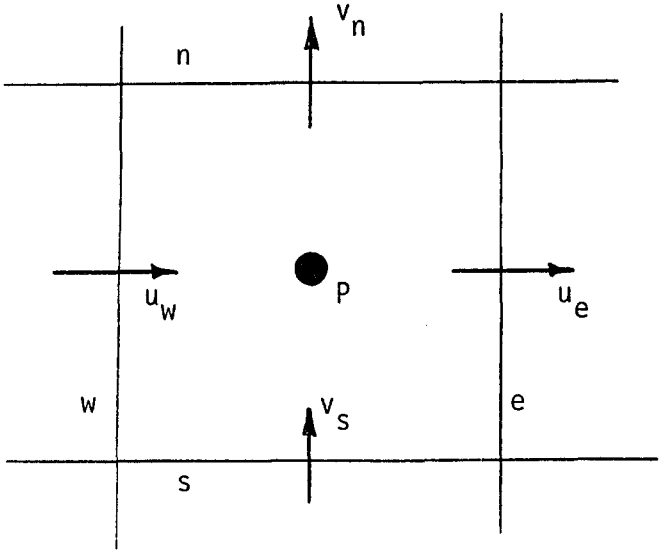


Fig. 2.1a A typical discretized domain for the general variable, ϕ.

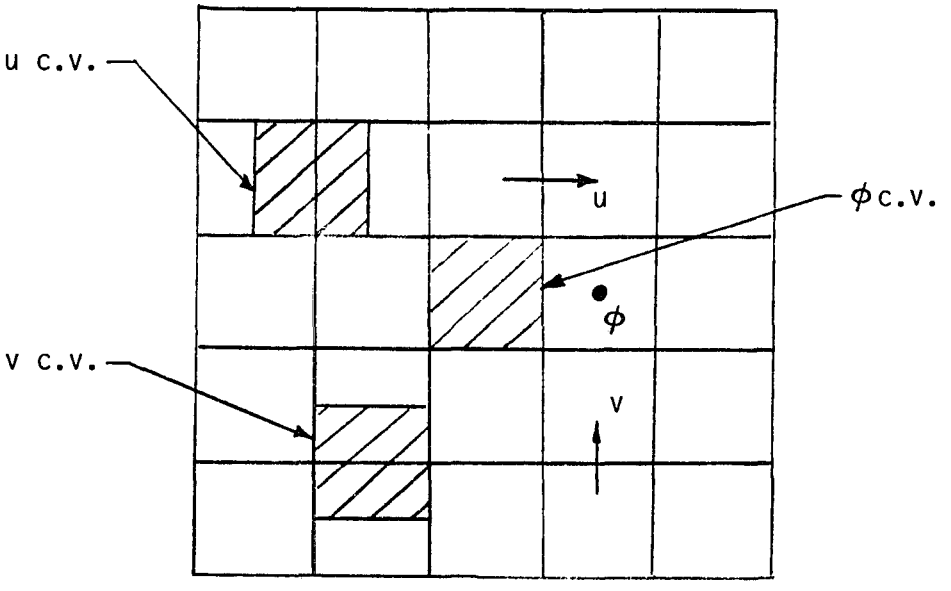


Fig. 2.1b Control volumes for velocity variables and scalar variables.

It should be noted that the third dimension is assumed to be one unit so that the control volume faces (lines) represent areas, while the control volume areas are actually volumes.

In this chapter, the flow field is assumed to be known. However, in general, the flow field is unknown, and the momentum and mass conservation equations must be solved. In solving these equations, a staggered grid, such as that shown in Fig. 2.1b, is usually employed to eliminate the possibility of predicting checkerboard pressure and velocity fields (see Patankar, 1982). These unrealistic fields may occur since only the first derivative of pressure appears in the momentum equations. Although the pressure values are implicitly specified by the continuity constraint, pressure itself does not appear explicitly in any of the governing equations. When pressure and velocity are stored at the same locations (e.g., the grid points in Fig. 2.1a) and central differences are employed to discretize the first order derrivatives of pressure in the momentum equations and velocity in the continuity equation, then the pressure and velocity differences between every other grid point rather than between adjacent grid points appear in the discretized system of equations. Thus, the momentum equations are insensitive to any differences between a uniform or a checkerboard pressure field. Similarly, the continuity equation is satisfied equally by both uniform and checkerboard velocity fields. These physically unrealistic checkerboard fields are

eliminated by using the staggered grid arrangement shown in Fig. 2.1b. Note that the u- and v-velocity values are stored at the control volume faces of the dependent variable ϕ. In this manner, pressure and velocity differences between adjacent rather than alternate grid points are employed in the discretized system of equations, preventing checkerboard pressure and velocity fields.

2.2 DIFFUSIVE TERM DISCRETIZATION

As previously stated, Eq. 2.1 is discretized using the control volume approach. In this approach, conservation is sought over each control volume, thus ensuring that conservation is always satisfied over any number of control volumes. The above equation is then integrated over each control volume identified by its grid point P, its grid point neighbors (N, S, E, and W) and the control volume faces (n, s, e, and w) in the north, south, east, and west directions, respectively (see Fig. 2.2).

The evaluation of the integral equation is accomplished through piecewise profiles which define the variation of the dependent variable ϕ between nodes. Different profiles may be used to approximate different parts of the integral (i.e., different integrands). The resulting discretized algebraic equation represents conservation over any given finite control volume.

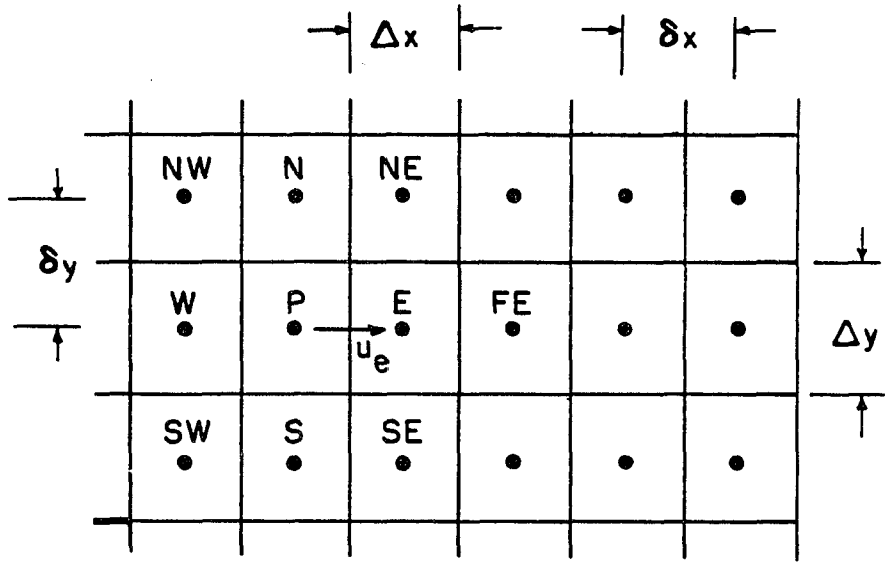


Fig. 2.2 A typical control volume and surrounding neighboring control volumes.

For the diffusive terms, a piecewise, linear profile for the variable ϕ is assumed so that

$$\iint_{cv} \frac{\partial}{\partial x}\left(\Gamma \frac{\partial \phi}{\partial x}\right) dx\,dy = \Gamma_e \frac{\Delta y}{\delta x_e}\left(\phi_E - \phi_P\right) - \Gamma_w \frac{\Delta y}{\delta x_w}\left(\phi_P - \phi_W\right) \qquad 2.2$$

and, similarly,

$$\iint_{cv} \frac{\partial}{\partial y}\left(\Gamma \frac{\partial \phi}{\partial y}\right) dx\,dy = \Gamma_n \frac{\Delta y}{\partial y_n}\left(\phi_N - \phi_P\right) - \Gamma_s \frac{\Delta x}{\partial y_s}\left(\phi_P - \phi_S\right) \qquad 2.3$$

Next the source term of Eq. 2.1 will be discretized. Note that the diffusive and source terms are discretized in exactly the same manner for both the first and third order upwind schemes.

2.3 SOURCE TERM DISCRETIZATION

The source term, S, in Eq. 2.1 may be a constant or some function of the dependent variable, ϕ. Since the set of equations generated by discretizing Eq. 2.1 will be solved using a method for linear algebraic equations, the source term is "linearized" by employing the following relationship,

$$S = S_c + S_P \phi_P \qquad 2.4$$

where S_c is the constant part of the source term and S_P is the coefficient of ϕ_P (Patankar, 1980).

Next the profile assumptions for the convective fluxes are examined. Two methods for discretizing the convective fluxes are presented. These are the simple, first order upwind approximation and a third order accurate upwind scheme

developed by Leonard (1979) called QUICK. The first order upwind scheme is examined first.

2.4 FIRST ORDER ACCURATE UPWIND SCHEME FOR CONVECTION TERMS

In the upwind scheme, the discretized form of the convective term can be written as

$$\iint_{cv} \frac{\partial}{\partial x}(\rho u\phi)\,dxdy=\Delta y[(\rho u\phi)_e-(\rho u\phi)_w] \qquad 2.5$$

and

$$\iint_{cv} \frac{\partial}{\partial y}(\rho v\phi)\,dxdy=\Delta x[(\rho u\phi)_n-(\rho u\phi)_s] \qquad 2.6$$

with

$$\phi_e = \phi_P \qquad for\ u_e \geq 0 \qquad 2.7a$$

and

$$\phi_e = \phi_E \qquad for\ u_e < 0 \qquad 2.7b$$

The interface value of ϕ at the other control volume faces (n, s and w) is evaluated similarly. The resulting discretized equation is of the form

$$a_P\phi_P = a_E\phi_E + a_W\phi_W + a_N\phi_N + a_S\phi_S + b \qquad 2.8a$$

or

$$a_P\phi_P = \sum a_{nb}\phi_{nb} + b \qquad 2.7b$$

where the summation (Σ) is over the neighboring (nb) grid points. The coefficients in the algebraic equation are defined as follows:

$$a_E = D_e + [-F_e, 0], \qquad 2.9a$$

$$a_W = D_W + [F_W, 0], \qquad 2.9b$$

$$a_N = D_n + [F_n, 0], \qquad 2.9c$$

$$a_S = D_s + [F_s, 0], \qquad 2.9d$$

$$a_P = a_E + a_W + a_N + a_S - S_P \Delta x \Delta y, \qquad 2.9e$$

$$b = S_C \Delta x \Delta y, \qquad 2.9f$$

where

$$D_e = \frac{\Gamma_e \, \Delta y}{(\delta x)_e}, \qquad D_w = \frac{\Gamma_w \, \Delta y}{(\delta x)_w},$$

$$D_n = \frac{\Gamma_n \, \Delta x}{(\delta y)_n}, \qquad D_s = \frac{\Gamma_s \, \Delta x}{(\delta y)_s},$$

and

$$F_e \equiv (\rho u)_e \, \Delta y, \qquad F_w \equiv (\rho u)_w \, \Delta y,$$

$$F_n \equiv (\rho v)_n \, \Delta x, \qquad F_s \equiv (\rho v)_s \, \Delta x.$$

In Eq. 2.9 the notation [A,B] denotes the larger of the two values. If $\underline{L}\phi = f$ is used to denote the differential equation where $\underline{L}$ is the differential operator, then Eq. 2.7b can be rewritten as

$$L\phi = a_P \phi_P - \Sigma a_{nb} \phi_{nb} = b \qquad 2.10$$

where L is the discretized differential operator.

2.5 THIRD ORDER ACCURATE QUICK SCHEME FOR CONVECTION TERMS

For the higher-order method, the QUICK scheme developed by Leonard (1988) is employed. In this method, the convective fluxes are discretized using a third-order accurate upwind scheme. For a uniform grid (see Fig. 2.2), the interface value ϕ_e is expressed in one of two manners depending upon the

value (sign) of the velocity perpendicular to the control volume face e.

If $u_e \geq 0$, then

$$\phi_e = \frac{(\phi_E+\phi_P)}{2} - \frac{(\phi_E-2\phi_P+\phi_W)}{8} + \frac{(\phi_N-2\phi_P+\phi_S)}{24} \qquad 2.11a$$

and if $u_e < 0$, then

$$\phi_e = \frac{(\phi_E+\phi_P)}{2} - \frac{(\phi_{FE}-2\phi_E+\phi_P)}{8} + \frac{(\phi_*-2\phi_E+\phi_{SE})}{24} \qquad 2.11b$$

Note that the interface value is composed of three terms, namely the central difference term, the normal curvature term and the transverse curvature term, respectively. The other interface ϕ values are evaluated similarly at the other control volume faces. The resulting equation can be written in a form identical to Eq. 2.8, with the coefficients defined differently.

The distribution of the normal and transverse curvature terms in Eq. 2.11 among the coefficients in Eq. 2.8 presents a potential problem. The system of equations may become unstable if diagonal dominance is lost. Han et al (1981) investigated this problem using the original version of Leonard's scheme (1979) without the transverse curvature terms, i.e.,

for $u_e \geq 0$,

$$\phi_e = \frac{(\phi_P+\phi_E)}{2} - \frac{(\phi_E-2\phi_P+\phi_W)}{8} \qquad 2.12a$$

and for $u_e < 0$,

$$\phi_e = \frac{(\phi_P + \phi_E)}{2} - \frac{(\phi_{FE} - 2\phi_E + \phi_P)}{8} \qquad 2.12b$$

As a first approach, they included the normal curvature terms in the explicit source term while distributing the central difference terms among the a_{nb} and a_p coefficients. Unfortunately, this obvious and seemingly innocuous strategy yielded an unstable scheme. They subsequently tried several other methods of distributing the terms and recommended the following:

for $u_e \geq 0$,

$$\phi_e = \frac{(6\phi_P + 4\phi_E)}{8} - \frac{(\phi_W + \phi_E)}{8} \qquad 2.13a$$

and, for $u_e < 0$

$$\phi_e = \frac{(3\phi_P + 4\phi_E)}{8} - \frac{(\phi_{FE} - 2\phi_E)}{8} \qquad 2.13b$$

where the first term is incorporated in the a_{nb} and a_p coefficients and the second term is included in the source term, b.

In this research, the decomposition recommended by Han, et al (1981) is employed, but with the further addition of the transverse curvature terms to the explicit source term, b. This scheme is stable when sufficiently underrelaxed. Thus, the resulting discretized form of Eq. 2.1 using the QUICK scheme is as follows:

$$a_P\phi_P = a_E\phi_E + a_W\phi_W + a_N\phi_N + a_S\phi_S + b. \qquad 2.14$$

If the following function definitions are made:

for $u_e \geq 0$	IPOSE=1 INEGE=0	for $u_e < 0$	IPOSE=0 INEGE=1
for $u_w \geq 0$	IPOSW=1 INEGW=0	for $u_w < 0$	IPOSW=0 INEGW=1
for $v_n \geq 0$	IPOSN=1 INEGN=0	for $v_n < 0$	IPOSN=0 INEGN=1
for $v_s \geq 0$	IPOSS=1 INEGS=0	for $v_s < 0$	IPOSS=0 INEGS=1

$$\text{SGN(A)} = \begin{cases} +1 & \text{if } A \geq 0 \\ -1 & \text{if } A < 0 \end{cases}$$

then the coefficients in Eq. 2.13 may be defined as

$$a_E = -0.5F_e + D_e \qquad 2.15a$$

$$a_w = F_w\left(\frac{IPOSW}{8}\right)\left[4 + \frac{(\delta x_i)^2}{\Delta x_{i-1}}\left(\frac{1}{\delta x_{i-1}} + \frac{1}{\delta x_i}\right)\right] + F_w\left(\frac{INEGW}{8}\right)\left[4 - \frac{\delta x_i}{\Delta x_i}\right] + D_w \qquad 2.15b$$

$$a_n = -0.5F_n + D_n \qquad 2.15c$$

$$a_s = F_s\left(\frac{IPOSS}{8}\right)\left[4 + \frac{(\delta y_j)^2}{\Delta y_{j-1}}\left(\frac{1}{\delta y_{j-1}} + \frac{1}{\delta y_j}\right)\right] + F_s\left(\frac{INEGS}{8}\right)\left[4 - \frac{\delta y_j}{\Delta y_j}\right] + D_s \qquad 2.15d$$

$$a_P = -S_P \Delta x_i \Delta y_j + D_e + D_w + D_n + D_s - \left(\frac{F_w+F_s}{2}\right)$$

$$+ F_e\left(\frac{IPOSE}{8}\right)\left[4 + \frac{\delta x_{i+1}}{\Delta x_i}\left(\frac{\delta x_{i+1}}{\delta x_i} + 1\right)\right]$$

$$+ F_e\left(\frac{INEGE}{8}\right)\left(4 - \frac{\delta x_{i+1}}{\Delta x_{i+1}}\right) + F_n\left(\frac{INEGN}{8}\right)\left(4 - \frac{\delta y_{j+1}}{\Delta y_{j+1}}\right)$$

$$+ F_n\left(\frac{IPOSN}{8}\right)\left[4 + \frac{\delta y_{j+1}}{\Delta y_j}\left(\frac{\delta y_{j+1}}{\delta y_j} + 1\right)\right] \qquad 2.15e$$

$$b = S_c \Delta x_i \Delta y_j + F_e\left(\frac{IPOSE}{8}\right)\frac{(\delta x_{i+1})^2}{\Delta x_i}\left[\frac{\phi_{i-1,j}}{\delta x_i} + \frac{\phi_{i+1,j}}{\delta x_{i+1}}\right]$$

$$+ F_e\left(\frac{INEGE}{8}\right)\frac{(\delta x_{i+1})^2}{\Delta x_{i+1}}\left[\frac{\phi_{i+2,j}}{\delta x_{i+2}} - \phi_{i+1,j}\left(\frac{1}{\delta x_{i+2}} + \frac{1}{\delta x_{i+1}}\right)\right]$$

$$- F_w\left(\frac{IPOSW}{8}\right)\frac{(\delta x_i)^2}{\Delta x_{i-1}}\left[\frac{\phi_{i-2,j}}{\delta x_{i-1}} + \frac{\phi_{i,j}}{\delta x_i}\right]$$

$$- F_w\left(\frac{INEGW}{8}\right)\frac{(\delta x_i)^2}{\Delta x_i}\left[\frac{\phi_{i+1,j}}{\delta x_{i+1}} - \phi_{i,j}\left(\frac{1}{\delta x_i} + \frac{1}{\delta x_{i+1}}\right)\right]$$

$$+ F_n\left(\frac{IPOSN}{8}\right)\frac{(\delta y_{j+1})^2}{\Delta y_j}\left[\frac{\phi_{i,j-1}}{\delta y_j} + \frac{\phi_{i,j+1}}{\delta y_{j+1}}\right]$$

$$+ F_n\left(\frac{INEGN}{8}\right)\frac{(\delta y_{j+1})^2}{\Delta y_{j+1}}\left[\frac{\phi_{i,j+2}}{\delta y_{j+2}} - \phi_{i,j+1}\left(\frac{1}{\delta y_{j+1}} + \frac{1}{\delta y_{j+2}}\right)\right]$$

$$- F_s\left(\frac{IPOSS}{8}\right)\frac{(\delta y_j)^2}{\Delta y_{j-1}}\left[\frac{\phi_{i,j-2}}{\delta y_{j-1}} + \frac{\phi_{i,j}}{\delta y_j}\right]$$

$$- F_s\left(\frac{INEGS}{8}\right)\frac{(\delta y_j)^2}{\Delta y_j}\left[\frac{\phi_{i,j+1}}{\delta y_{j+1}} - \phi_{i,j}\left(\frac{1}{\delta y_j} + \frac{1}{\delta y_{j+1}}\right)\right]$$

$$- F_e\left(\frac{IPOSE}{24}\right)\Delta y_j\left[\frac{\phi_{i,j+1}}{\delta y_{j+1}} + \frac{\phi_{i,j-1}}{\delta y_j} - \phi_{i,j}\left(\frac{1}{\delta y_j} + \frac{1}{\delta y_{j+1}}\right)\right]$$

$$- F_e\left(\frac{INEGE}{24}\right)\Delta y_j\left[\frac{\phi_{i+1,j+1}}{\delta y_{j+1}} + \frac{\phi_{i+1,j-1}}{\delta y_j} - \phi_{i+1,j}\left(\frac{1}{\delta y_j} + \frac{1}{\delta y_{j+1}}\right)\right]$$

$$+ F_w\left(\frac{IPOSW}{24}\right)\Delta y_j\left[\frac{\phi_{i-1,j+1}}{\delta y_{j+1}} + \frac{\phi_{i-1,j-1}}{\delta y_j} - \phi_{i-1,j}\left(\frac{1}{\delta y_j} + \frac{1}{\delta y_{j+1}}\right)\right]$$

$$+ F_w\left(\frac{INEGW}{24}\right)\Delta y_j\left[\frac{\phi_{i,j+1}}{\delta y_{j+1}} + \frac{\phi_{i,j-1}}{\delta y_j} - \phi_{i,j}\left(\frac{1}{\delta y_j} + \frac{1}{\delta y_{j+1}}\right)\right]$$

$$- F_n\left(\frac{IPOSN}{24}\right)\Delta x_i\left[\frac{\phi_{i+1,j}}{\delta x_{i+1}} + \frac{\phi_{i-1,j}}{\delta x_i} - \phi_{i,j}\left(\frac{1}{\delta x_i} + \frac{1}{\delta x_{i+1}}\right)\right]$$

$$- F_n\left(\frac{INEGN}{24}\right)\Delta x_i\left[\frac{\phi_{i+1,j+1}}{\delta x_{i+1}} + \frac{\phi_{i-1,j+1}}{\delta x_i} - \phi_{i,j+1}\left(\frac{1}{\delta x_i} + \frac{1}{\delta x_{i+1}}\right)\right]$$

$$+ F_s\left(\frac{IPOSS}{24}\right)\Delta x_i\left[\frac{\phi_{i+1,j-1}}{\delta x_{i+1}} + \frac{\phi_{i-1,j-1}}{\delta x_i} - \phi_{i,j-1}\left(\frac{1}{\delta x_i} + \frac{1}{\delta x_{i+1}}\right)\right]$$

$$+ F_s\left(\frac{INEGS}{24}\right)\Delta x_i\left[\frac{\phi_{i+1,j}}{\delta x_{i+1}} + \frac{\phi_{i-1,j}}{\delta x_i} - \phi_{i,j}\left(\frac{1}{\delta x_i} + \frac{1}{\delta x_{i+1}}\right)\right] \qquad 2.15f$$

A brief derivation of Eqs. 2.14 and 2.15 is presented in the Appendix.

At the boundaries, Leonard (1988) recommends the use of psuedonodes outside the physical domain as a convenient tool to retain the general form of Eq. 2.14. The psuedonode values may be calculated by quadratic extrapolation assuming locally one-dimensional quadratic behavior normal to the boundary and using the known boundary conditions along with the most current values of the adjacent physical nodes. In the present work, the flagged regions will generally have two or more

boundaries in the domain interior. In such cases, the "psuedonodes" are actual physical nodes where the upwind solution is available (since the upwind solution is obtained in the entire domain prior to calculating the solution in the flagged regions). Thus, for the interior boundaries of the flagged region, the upwind solution along the flagged boundary and at the adjacent nodes is used in implementing the boundary condition.

2.6 SOLUTION PROCEDURE

The typical solution procedure employs an iterative method in which the coefficients for Eq. 2.8 or 2.14 are calculated in each iteration and a line-by-line Thomas algorithm is used to solve for the values of the dependent variable, ϕ. The updated ϕ values are then used in the next iterate coefficient calculation and the scheme is repeated until convergence.

Equation 2.8 or Equation 2.14 is the typical discretized equation for internal grid points. The discretized form of Eq. 2.1 for the boundary control volumes depends upon the boundary conditions given. If the value of the dependent variable is known at a boundary, e.g., $\phi = \phi_0$, then this is the algebraic equation for the boundary point. If, however, the flux is known at the boundary, then its value can be used directly in Eq. 2.1 and only the remaining fluxes along the other three sides of the boundary control volume are

discretized. In this manner, the number of algebraic equations is always equal to the number of unknowns. Hence, the problem is mathematically well defined.

The system of algebraic equations to be solved is nonlinear since the coefficients depend upon Γ which may be itself a function of the dependent variables. Also, the source term, b, may be dependent on ϕ. With these two sources of nonlinearity an iterative process is better suited to the problem than a direct solution. The method employed in this research is the line-by-line Tri-diagonal Matrix Algorithm (TDMA). The TDMA is a direct method for one-dimensional problems, and is applicable to two-dimensional problems when applied repeatly along lines of constant x or y. During these sweeps (parallel to either the x- or y-axis), points not lying along the line on which the solution is sought are treated as known values (with their most recent values being used). The convergence of this method, which is faster than that of point-by-point methods, is accelerated by applying block correction (see Prakash and Patankar, 1981). For highly nonlinear problems, underrelaxation may be required.

The solution procedure used to solve the resulting set of algebraic equations for convection-diffusion problems is as follows:

1. Start with a guessed value of the dependent variable, ϕ, at all internal nodes.

2. Use ϕ calculated in the previous iteration (or the guessed ϕ for the first iteration) to update the coefficients and the source term.
3. Solve the linearized system of equations by an iterative method (line-by-line TDMA) to obtain a new ϕ field.
4. Go back to Step 2 and repeat until a solution with the desired convergence level is obtained.

2.7 TEST PROBLEMS

Two convection-diffusion problems are studied. The first problem considered is conduction in a rotating hollow cylinder, while the second problem studied is the standard transport of a step change of a scalar variable. The third-order accurate QUICK solution is compared with the exact solution and with the first-order upwind solution on the same mesh.

2.7.1 Radial Heat Condution in a Rotating Hollow Cylinder

This problem is a standard one for testing numerical convection-diffusion schemes (see, for example, Hsu (1981) and Moukalled (1987)). As shown in Fig. 2.2, the hollow cylinder has an inner radius of R_i and outer radius $3R_i$. The temperatures on the inner and outer radii are known and given as t_i and t_o, respectively. Also known are the constant values

for the cylinder's angular velocity ω, the density ρ, specific heat c_p and thermal conductivity k.

The temperature field in the squared domain shaded in Fig. 2.2 is governed by the equation

$$U\frac{\partial T}{\partial X} + V\frac{\partial T}{\partial Y} = \frac{1}{P_e}\left(\frac{\partial^2 T}{\partial X^2} + \frac{\partial^2 T}{\partial y^2}\right) \qquad 2.16$$

where the dimensionless variables used are defined as follows:

$$X = \frac{x}{R_i}, \qquad Y = \frac{y}{R_i},$$

$$U = \frac{u}{\omega R_i}, \qquad V = \frac{v}{\omega R_i},$$

$$T = \frac{t - t_o}{t_i - t_o},$$

and the Peclet number is defined as

$$Pe = \frac{\rho \omega^2 R_i^2 C_p}{k}.$$

This problem has the following exact solution:

$$U = 2Y, \qquad 2.17a$$

$$V = -2X, \qquad 2.17b$$

$$T = 1 - \frac{\ln(X^2+Y^2)}{2\ln(3)} \qquad 2.17c$$

The percentage error of each numerical scheme is then calculated by

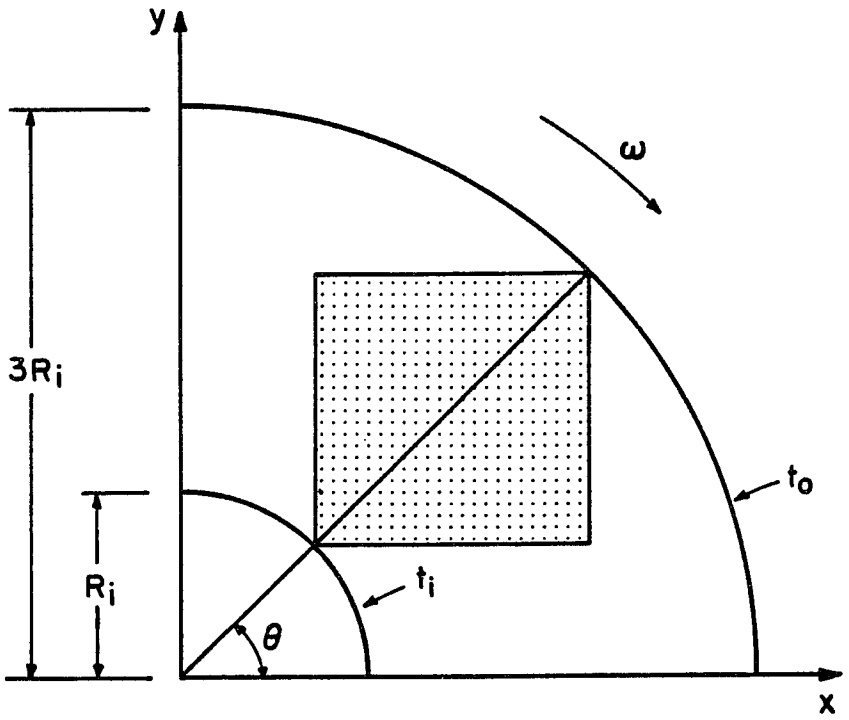


Fig. 2.3 Physical domain for the radial conduction in a rotating hollow cylinder.

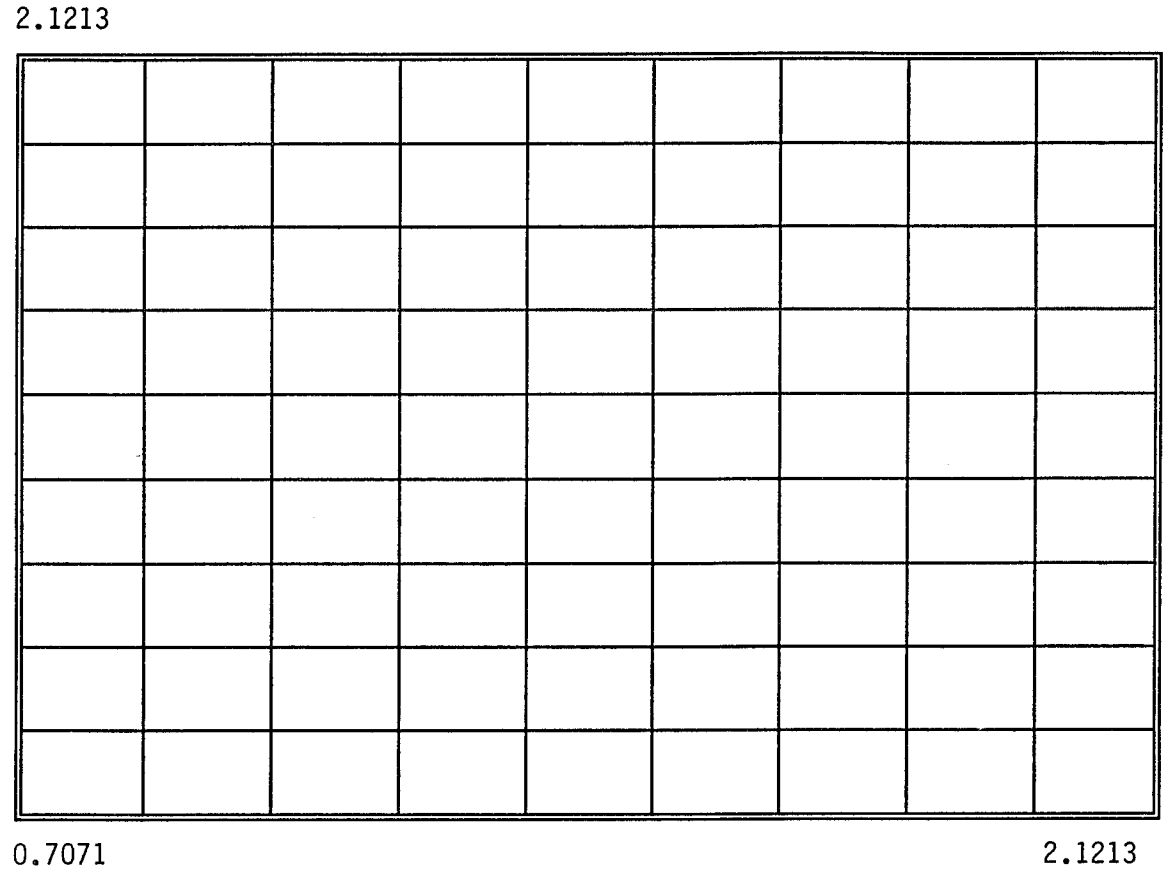


Fig. 2.4 Discretized physical domain for radial conduction in a rotating hollow cylinder.

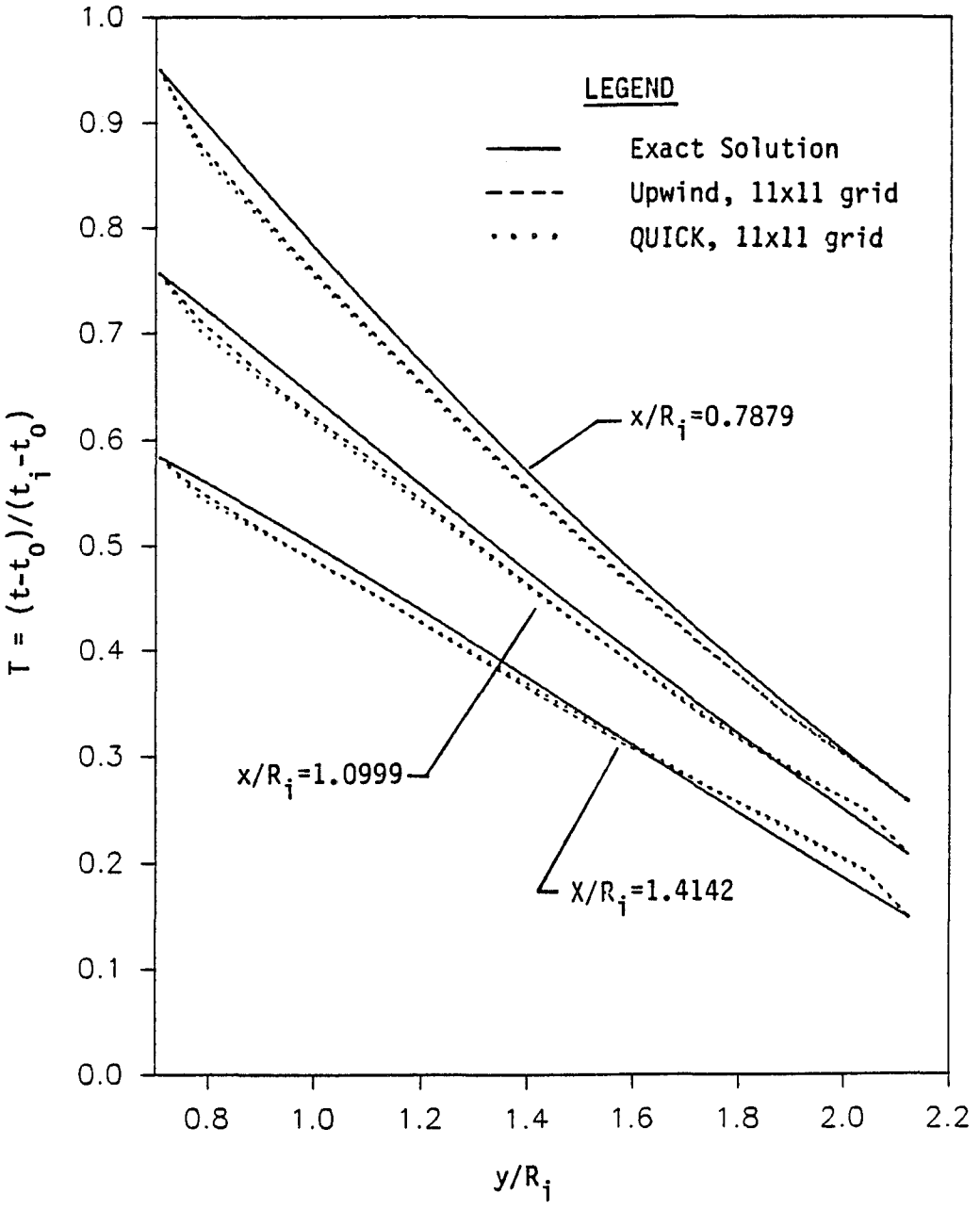


Fig. 2.5 Dimensionless temperature profile at x/R_i = 0.7857, 1.0999 and 1.4142 for radial conduction in a rotating hollow cylinder. Comparing upwind, QUICK and exact solutions.

$$\%Error = 100\left(\frac{T_{exact} - T_{computed}}{T_{exact}}\right) \qquad 2.18$$

The correct velocity field and boundary temperatures are given as input. A value of 100 for the Peclet number is used in obtaining the solution. The problem is solved on an 11 x 11 mesh. The discretized domain for the 11 x 11 grid is shown in Fig. 2.4.

Figure 2.5 shows the dimensionless temperature profile as a function of y/R_i at three different x/R_i locations. This figure compares the first order upwind solution, the third order QUICK solution and the exact solution. At all three x/R_i locations, the QUICK solution is closer to the exact solution as may be expected, since it is a higher order scheme. At x/R_i = 0.7879 the maximum percent error for the upwind solution is 3.83% while it is only 3.15% for the QUICK scheme. The maximum percent errors are 5.72% and 6.19% for the upwind and QUICK schemes, respectively, at x/R_i = 1.0999. Finally, at x/R_i = 1.4142, the maximum percent errors are 11.70% for the upwind method and 11.04% for the QUICK solution.

2.7.2 Transport of a Step Change of a Scalar Variable

This problem is also a standard test case for convection-diffusion problems (see, for example, Leschziner (1980), Hsu (1981), and Moukalled (1987)). For the physical situation depicted in Fig. 2.6, L is the length of the square domain while u_{fs} and v_{fs} are the free stream velocities in the x- and

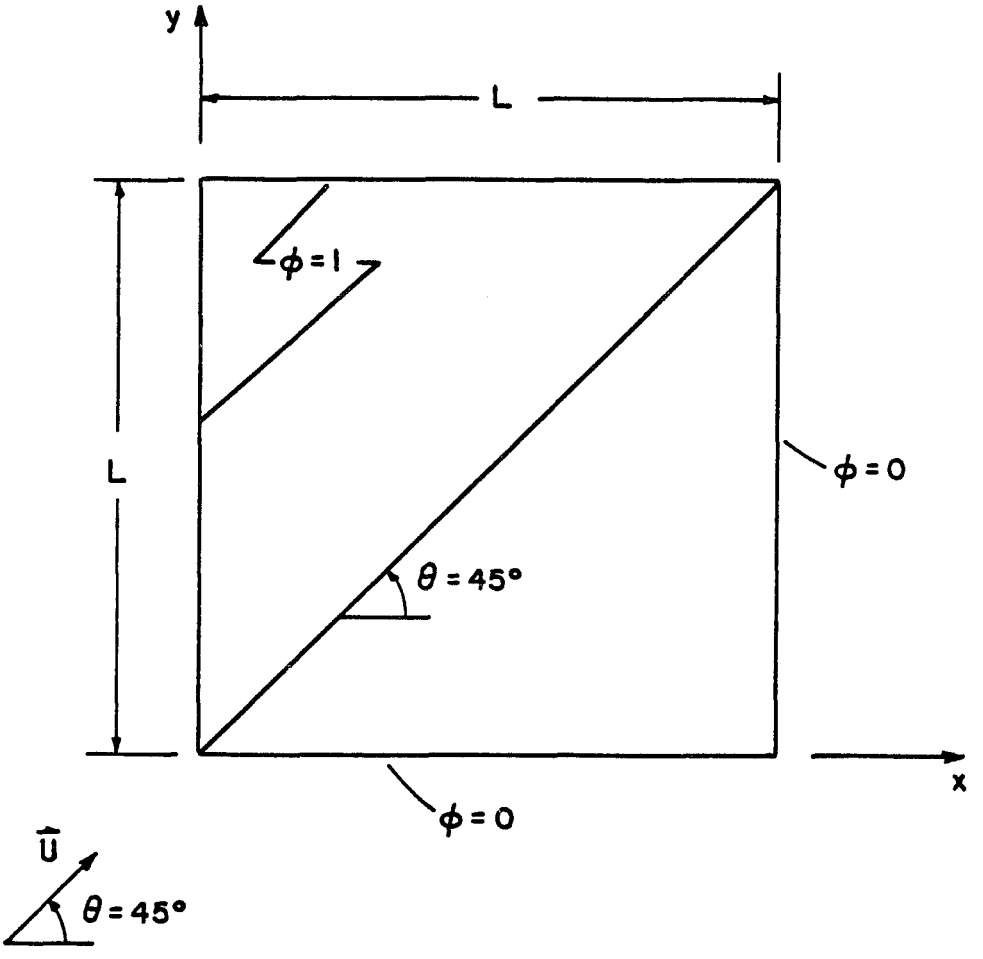


Fig. 2.6 Physical domain and boundary conditions for the transport of a step change of a scalar variable in a region with a uniform velocity field.

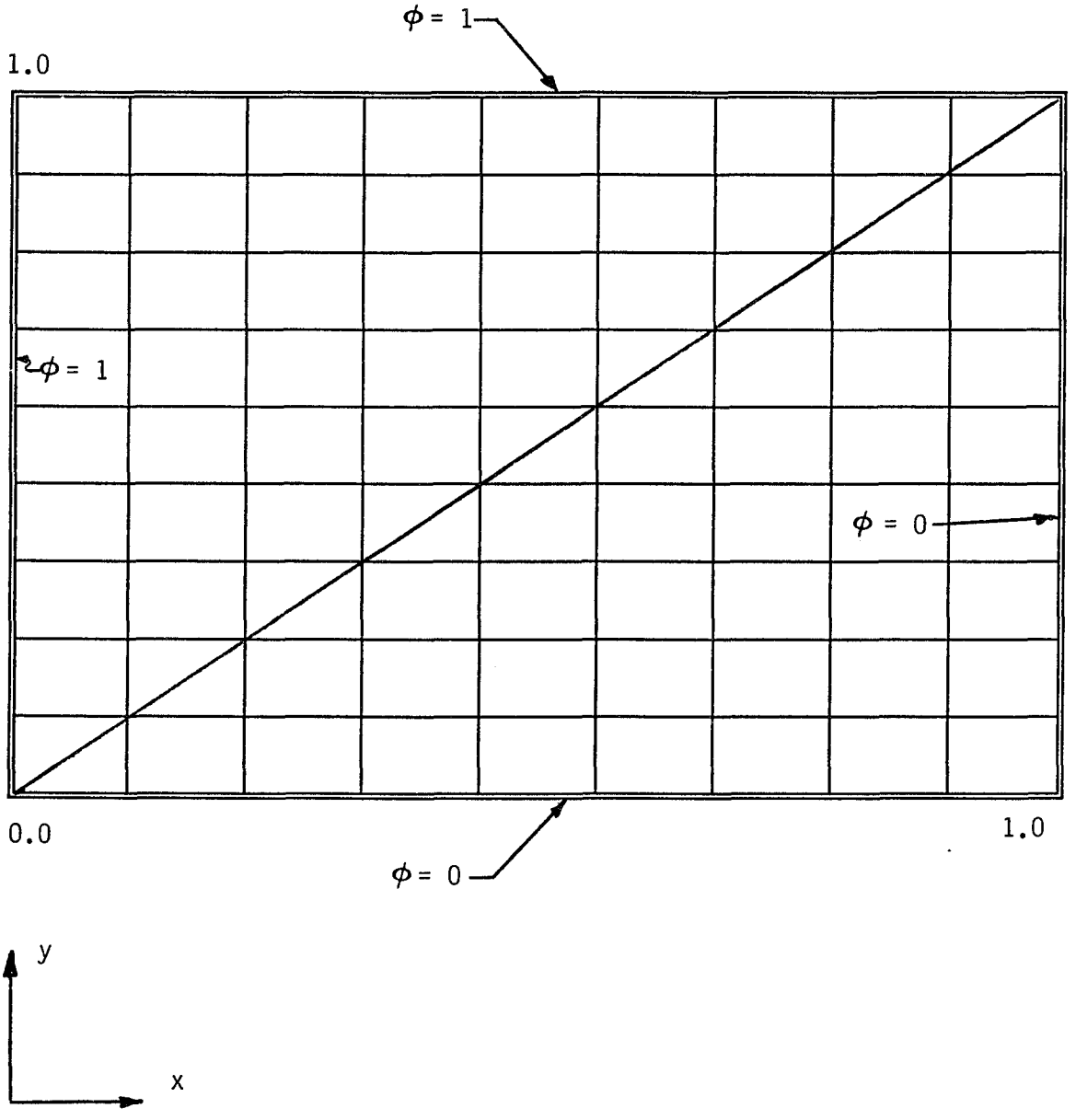


Fig. 2.7 Discretized physical domain and flagged region for the transport of a step change of a scalar variable in a region with a uniform velocity field.

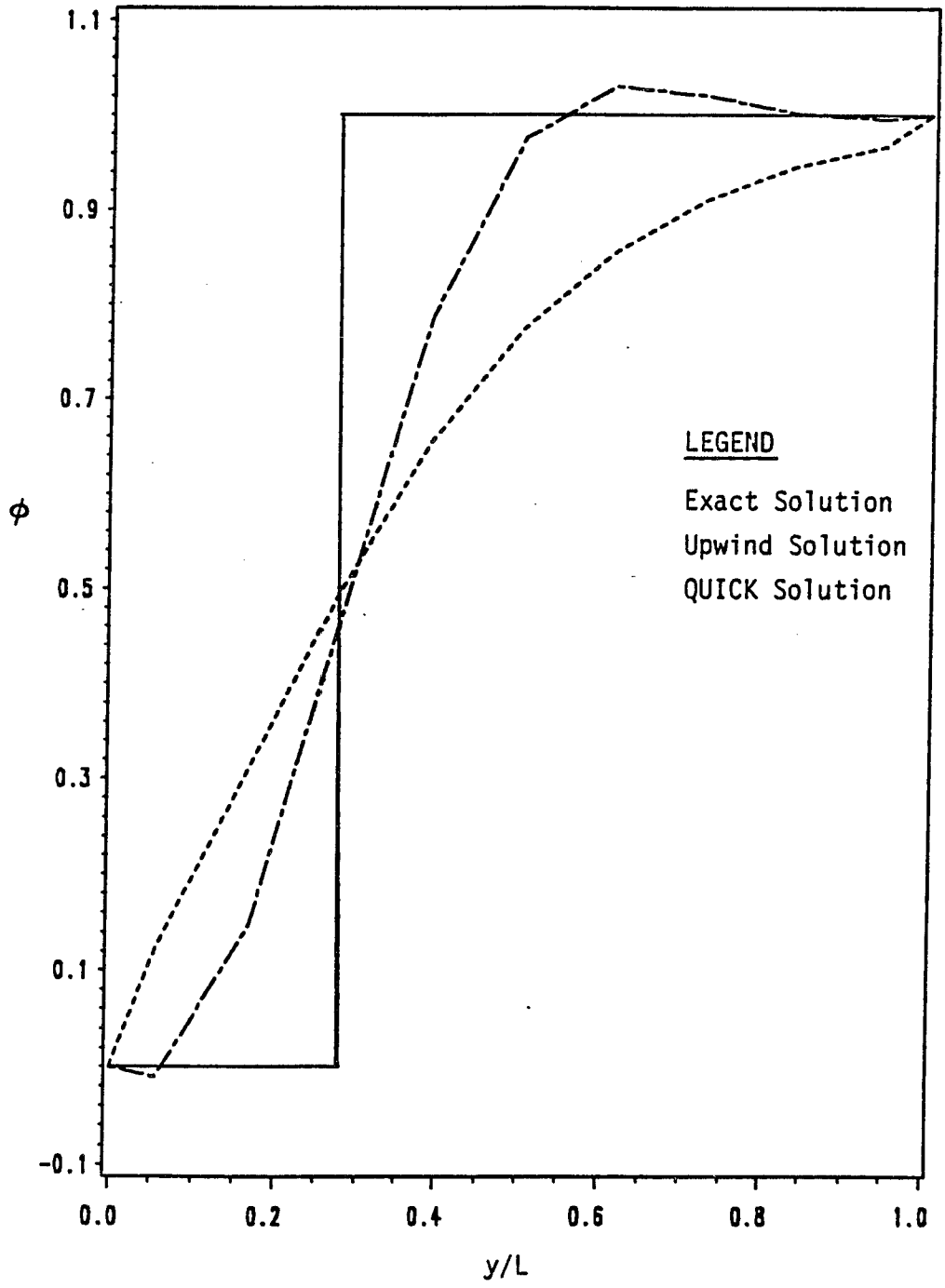


Fig. 2.8 ϕ profile at x/L = 0.2778 for the transport of a step change of a scalar variable in a region with a uniform velocity field. Comparing upwind, QUICK and exact solutions.

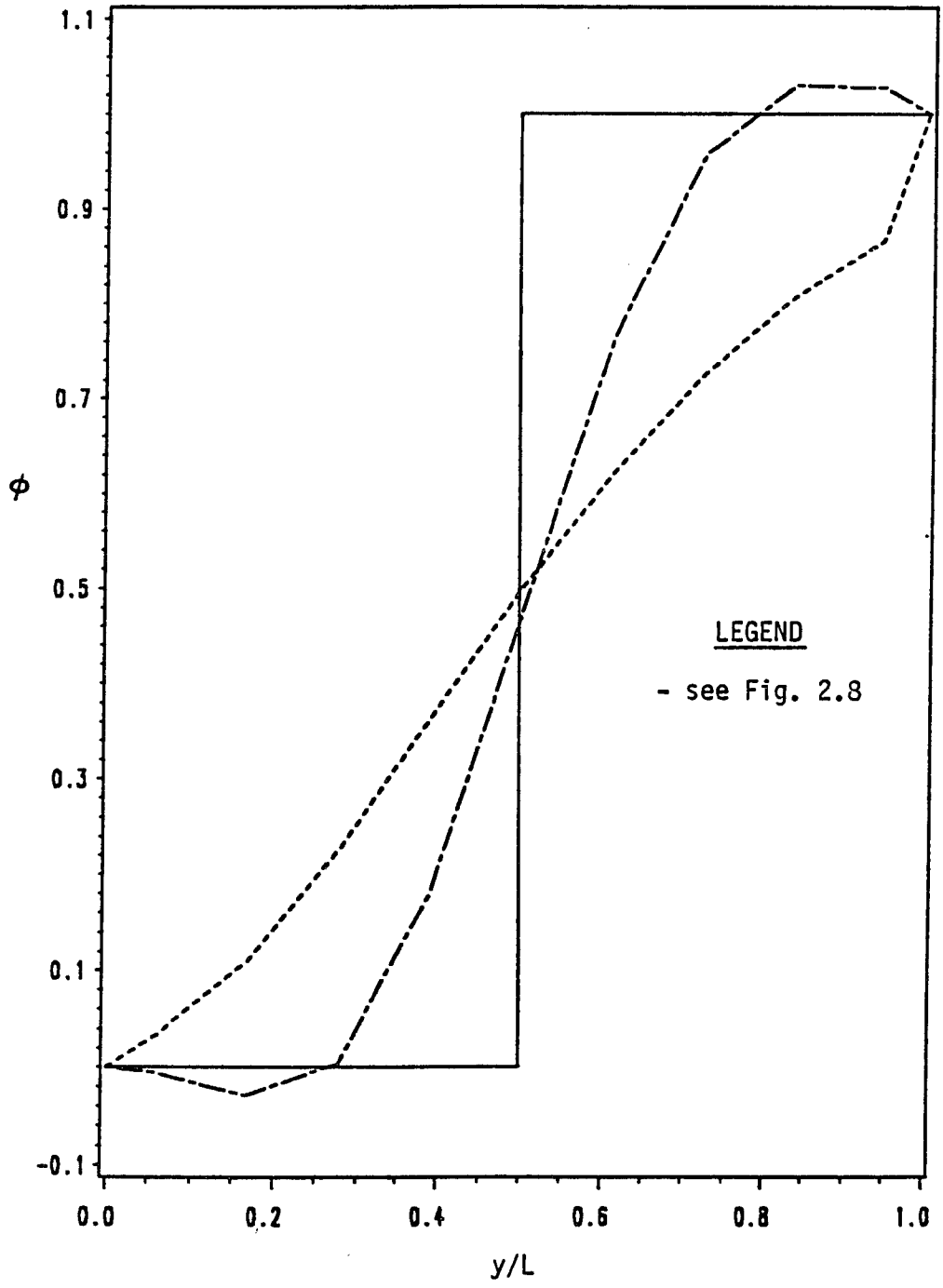


Fig. 2.9 ϕ profile at x/L = 0.5000 for the transport of a step change of a scalar variable in a region with a uniform velocity field. Comparing upwind, QUICK and exact solutions.

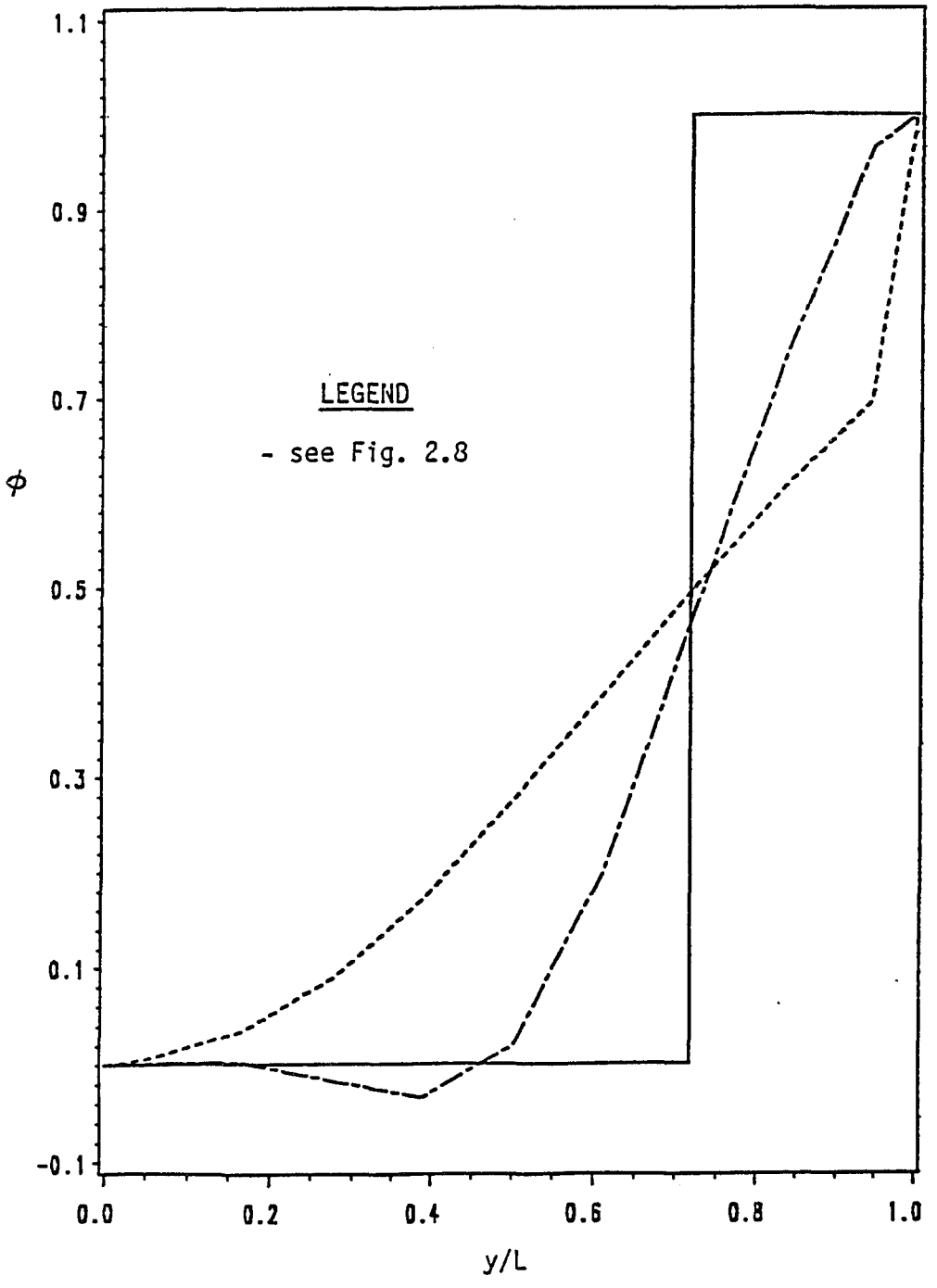


Fig. 2.10 ϕ profile at x/L = 0.7222 for the transport of a step change of a scalar variable in a region with a uniform velocity field. Comparing upwind, QUICK and exact soluttions.

y-directions, respectively. Using the dimensionless variables,

$$U = u/u_{fs}, \qquad V = v/u_{fs},$$

$$X = x/L, \qquad Y = y/L,$$

the problem is described by the equation

$$\frac{\partial}{\partial X}(U\phi) + \frac{\partial}{\partial Y}(V\phi) = \Gamma\left(\frac{\partial^2\phi}{\partial X^2} + \frac{\partial^2\phi}{\partial Y^2}\right) \qquad 2.19$$

If the diffusion coefficient, Γ, is set equal to zero, the artificial diffusion associated with each of the numerical schemes may be examined. The exact solution for this problem with $\Gamma = 0$ is (see Fig. 2.6)

$$\phi = 1 \quad \textit{above the } \Theta = 45^\circ \textit{ line}, \qquad 2.20a$$

$$\phi = 0 \quad \textit{below the } \Theta = 45^\circ \textit{ line} \qquad 2.20b$$

The numerical error for the finite difference approximations may be computed as

$$Error = |\phi_{exact} - \phi_{computed}|. \qquad 2.21$$

The discretized domain is shown in Fig. 2.7 for the 11 x 11 grid. Figs. 2.8-2.10 compare the results obtained using the upwind scheme and the QUICK method with the exact solution for several values of x/L. The QUICK solution overshoots the exact solution near y/L = 0.6 by 3% for x/L = 0.2778. At x/L = 0.5000, the QUICK solution again overshoots the exact solution by 2.3% at y/L = 0.9. The QUICK solution undershoots the exact solution for x/L = 0.722 by 3.5% at y/L = 0.95.

These small overshoots and undershoots are typical of a higher order scheme. In spite of these overshoots and undershoots, the QUICK scheme is still closer to the exact solution at each value of x/L.

2.8 CLOSING REMARKS

The governing general differenital equation was discretized using two schemes - namely, the first order upwind and the third order QUICK schemes. The solution procedure for convection-diffusion problems was presented. Results for two test problems were examined and compared with the exact solution in each case.

Chapter Three introduces three adaptive differencing techniques.

CHAPTER THREE

ADAPTIVE DIFFERENCING METHODS FOR CONVECTION-DIFFUSION PROBLEMS

Three adaptive differencing schemes are developed for convection-diffusion problems. In these schemes, regions of large errors are flagged as the solution evolves, and a higher order discretization scheme is used in the flagged regions. A multigrid approach is incorporated into the adaptive differencing procedure, and solutions are successively improved in the flagged and the unflagged regions until the desired accuracy levels are obtained. Solutions are obtained for two convection-diffusion problems and results with the adaptively differenced scheme show significant improvements.

3.1 SOLUTION PROCEDURE

In the three procedures developed herein, a first order upwind scheme is initially used to obtain values of ϕ over the entire domain until the following convergence criteria is met:

$$\left[\sum_{ij} D_{ij}^2\right]^{1/2} \sim 10^{-5} \qquad 3.1$$

where

$$D_{ij} = \frac{\phi_{ij}^{n} - \phi_{ij}^{n-1}}{1 + \phi_{ij}^{n}}.$$

Once the upwind solution has been obtained, regions of large error estimate are flagged, and thus a number of flagged regions are defined.

For the purpose of flagging points, the error estimate over the domain is claculated by

$$E = \alpha_1|\phi| + \alpha_2|\nabla\phi| + \alpha_3|\nabla^2\phi| \qquad 3.2$$

where α_1, α_2 and α_3 are constants which can be modified as required. In this research, the values employed for the coefficients are as follows: $\alpha_1 = 0$; $0.8 < \alpha_2 < 1.0$; $0.0 < \alpha_3 < 0.2$. Since points are flagged from the first order upwind solution, Eq. 3.2 with $\alpha_1 = 0$, $\alpha_2 \neq 0$ and $\alpha_3 \neq 0$ represents a reasonable approximation for the coefficients of the leading terms of the truncation error which (for a general first order scheme) can be written as

$$h|\nabla\phi| + \frac{h^2}{2!}\nabla^2\phi$$

where h is the mesh size. This error estimate is then normalized as

$$E_n = \frac{1.0 + E}{1.0 + E_{max}} \qquad 3.3$$

and grid points are flagged if the local value of E_n exceeds a predetermined threshold value ranging from 0.65 to 0.95. The flagged grid points are then organized into regions of

contiguously flagged points. These regions may or may not be rectangular in shape. One major advantage of the present method is that the flagged regions may be arbitrarily shaped without degrading the improved solution locally and without the additional computational expense of including unflagged grid points in a region or cluster in order to rectangularize the flagged area (as in Berger and Jameson, 1985).

To obtain a more accurate solution in these flagged regions(s), the quadratic upwind scheme, QUICK, is employed. The resulting set of discretized equations in both the flagged domain(s) and the global domain is solved using three different multiplegrid techniques. However, each of the three solution procedures employs an iterative method in which the coefficients for Eq. 2.7 and Eq. 2.13 are calculated in each iteration and a line-by-line Thomas algorithm is used to solve for the values of the dependent variable, ϕ. The updated ϕ values are then used in calculating the coefficients for the next iteration, and the scheme is repeated until the desired convergence level is achieved as in the procedure given in Chapter 2.

3.2 MULTIPLEGRID ADAPTIVE DIFFERENCING SCHEME 1 - WHOLE AND FLAGGED DOMAIN SWEEPS (MAD1 - WFDS)

An important aspect of the present procedure is the use of a multigrid calcuation strategy to drive the solution in both the flagged regions (inner grid, denoted by $\Omega_{l,i}$ with the

first subscript representing the refinement level and the second subscript representing each of the flagged regions) and the overall calculation domain (outer grid, denoted by Ω_0) to the desired accuracy level. The discretized outer grid equation, based on the first order upwind discretization, can be written as (Eq. 2.10)

$$L_0\phi_0 = b \qquad \text{in } \Omega_0 \tag{3.4}$$

where L is the discretized differential operator. The discretized inner grid equation, based on the third order QUICK scheme, is given by

$$L_1\phi_1 = a_P\phi_P - \Sigma a_{nb}\phi_{nb} = b \qquad \text{in } \Omega_{1,i}, \quad i=1,2,... \tag{3.5}$$

The subscript 0 denotes quantities on the outer grid and the subscript 1 denotes quantities on the inner grid.

The boundary conditions for Eq. 3.5 are based on ϕ_0, and therefore the solution accuracy in $\Omega_{1,i}$ depends not only on the discretization error associated with Eq. 3.4 but also on the accuracy of the boundary conditions. It is, therefore, important that the accuracy of the boundary conditions along the $\Omega_{1,i}$ boundaries be improved. To this end, a multigrid strategy is adopted, in which, after the solution in $\Omega_{1,i}$ is obtained, an improved solution is generated on the outer grid (Ω_0) by solving

$$L_0\phi_0 = b \qquad \text{in } (\Omega_0-\Omega_{1,i}) \tag{3.6a}$$

and

$$L_0\phi_0 = L_0\phi_1 \qquad \text{in } \Omega_{1,i} \tag{3.6b}$$

Equation 3.6a is identical to Eq. 2.10 while Eq. 3.6b may also be written as:

$$a_P^0 \phi_P^0 - \sum a_{nb}^0 \phi_{nb}^0 = a_P^0 \phi_P^1 - \sum a_{nb}^0 \phi_{nb}^1 \qquad 3.6c$$

where the superscript 0 indicates quantities computed using the upwind scheme and the superscript 1 denotes quantities calculated employing the QUICK scheme. Note that Eq. 3.6b forces ϕ_0 to be equal to ϕ_1 in $\Omega_{1,i}$, and this in turn improves the solution accuracy of the points in the unflagged regions $(\Omega_0 - \Omega_{1,i})$. The MAD1-WFDS algorithm, therefore, proceeds as follows:

1. In Ω_0, obtain the first order solution $L_0\phi_0 = b$.
2. Flag grid points with error estimates E_n greater than a specified tolerance, and define $\Omega_{1,i}$.
3. In $\Omega_{1,i}$, obtain the third order solution $L_1\phi_1 = b$ with boundary conditions based on ϕ_0.
4. In Ω_0, obtain improved solutions by solving

   $L_0\phi_0 = b$ in $(\Omega_0 - \Omega_{1,i})$

   $L_0\phi_0 = L_0\phi_1$ in $\Omega_{1,i}$.
5. Return to Step 3 and repeat until the desired accuracy level is satisfied.
6. Proceed to the next level of adaptive differencing by defining $\Omega_{2,i}$, which will

generally be nested in $\Omega_{1,i}$, and repeat above steps. Continue to the desired accuracy levels.

There are many similarities between this multiple-grid adaptive differencing scheme and a multigrid method. In a true multigrid method, an initial solution is obtained on a coarse grid and regions of high error estimate flagged (see, for example, Moukalled and Acharya, 1991) as in the above mulitple-grid method, MAD1-WFDS. At this point, however, the multigrid method generates a new finer mesh in the flagged region(s) rather than adopting a higher order differencing scheme. The solution is obtained on the finer grid. Then, an interpolating scheme is used to map the improved values of the dependent variables obtained on the finer grid back onto the grid points on the coarse mesh. These coarse grid improved values are used in the next solution iteration on the coarse grid. In this manner, the solution on the original domain is improved. Adaptation between the fine and coarse grids continues until the desired accuracy levels are achieved as in the MAD1-WFDS scheme.

Multigrid methods employ calculations performed on multiple overlaying grids with prolongation from the coarse grid to the fine grid and restriction from the fine grid to the coarse grid. The MAD1-WFDS scheme may be cast in this form. Generally the prolongation operation is achieved by some form of interpolation. In the MAD1-WFDS scheme, Step 3

above may be considered as a prolongation step. The boundary values ϕ_B^1 can be interpreted as $I_0^1\phi_0$ where I_0^1 is the prolongation operator in multigrid terminology. Step 4 is the corresponding restriction step. The $L_0\phi_1$ term can be interpreted as

$$b + I_1^0 \ (b^1 - L^1\phi_1)$$

where I_1^0 is the restriction operator.

Next, this multiple-grid scheme is applied to the two test case problems introduced in Chapter Two.

3.2.1 Radial Heat Conduction in a Rotating Hollow Cylinder

This problem statement was presented in Section 2.7.1 and is not repeated here. The problem is solved on an 11 x 11 mesh. The discretized domain and flagged region for the 11 x 11 grid is shown in Fig. 3.1.

Figures 3.2, 3.3, and 3.4 show the dimensionless temperature profile as a funciton of y/R_i at three different x/R_i locations. The first stage solution is the upwind solution in Ω_0, while the second and third stage solutions are the improved adaptively differenced outer grid solutions in Ω_0 after Step 4 of the multigrid algorithm described in the previous section. The second and third stage solutions are nearly identical and fall on top of each other in the plot. Clearly, the adaptively differenced solution is superior and closer to the exact solution than the first stage upwind solution. The flagged region (see Fig. 3.1) is indicated on

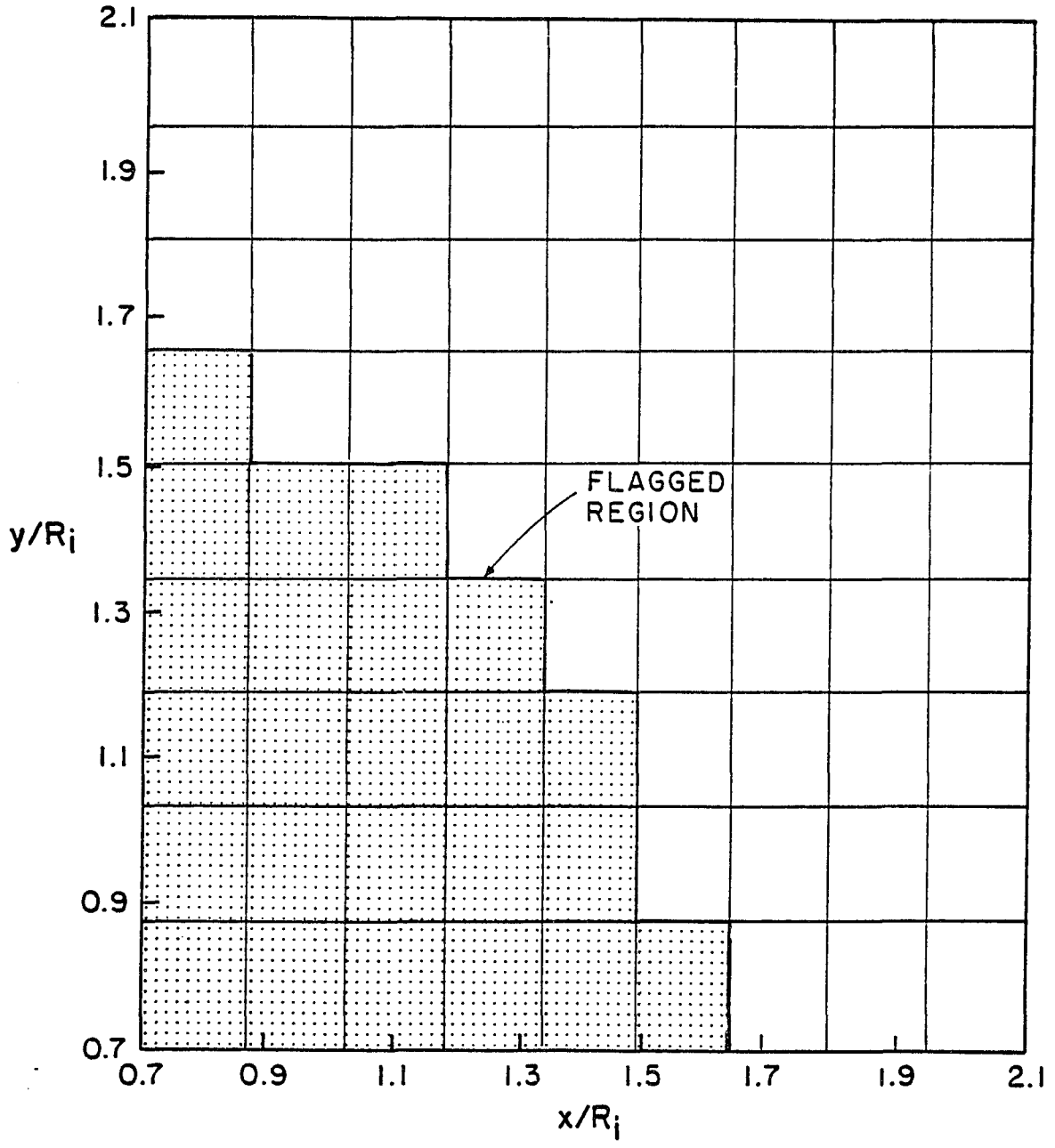


Fig. 3.1 Discretized physical domain and flagged region for radial conduction in a rotating hollow cylinder.

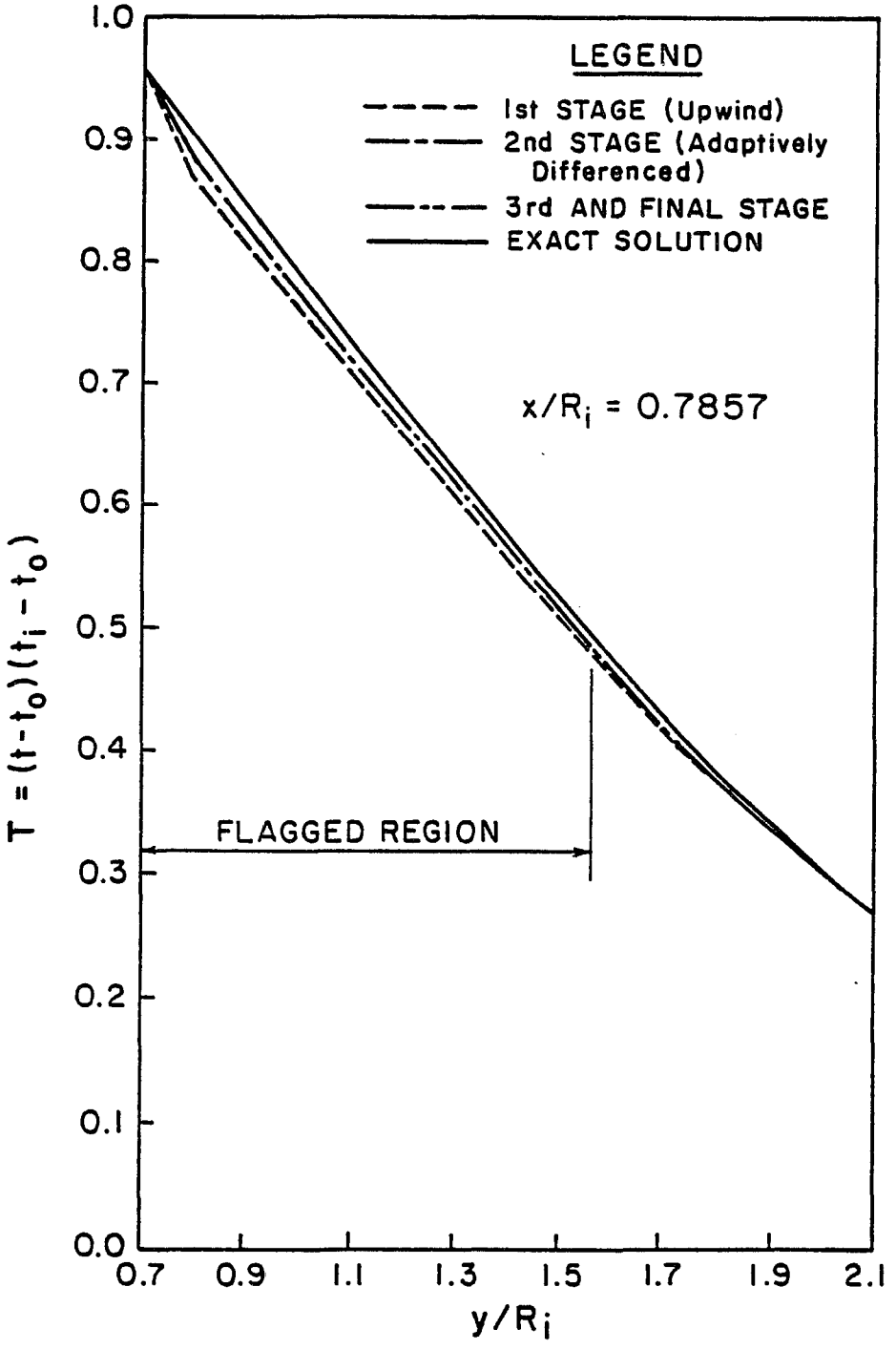


Fig. 3.2 Dimensionless temperature profiles at $x/R_i = 0.7857$ for radial conduction in a rotating hollow cylinder. Comparing upwind, MAD1-WFDS and exact solutions.

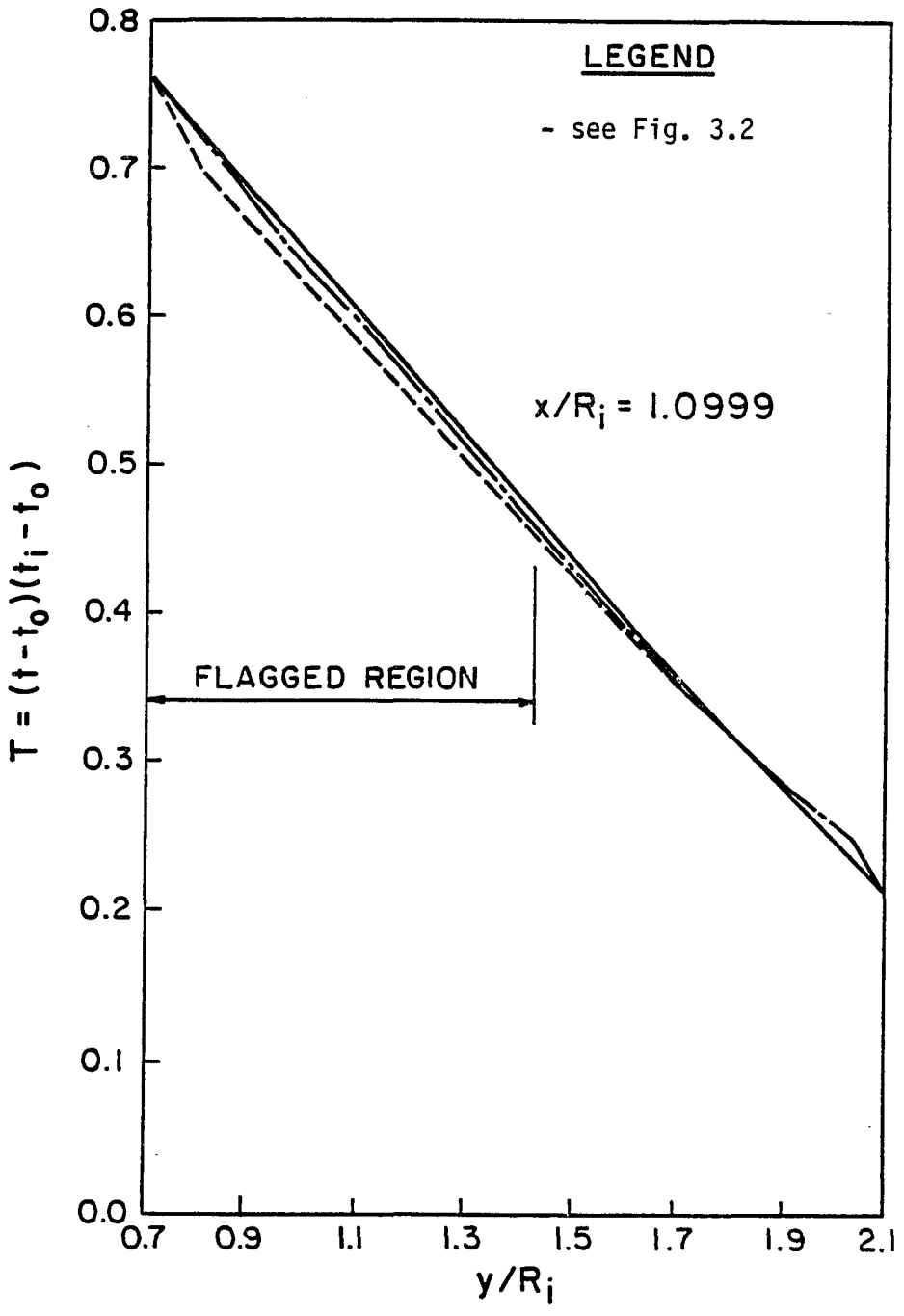


Fig. 3.3 Dimensionless temperature profile at $x/R_i = 1.0999$ for radial conduction in a rotating hollow cylinder. Comparing upwind, MAD1-WFDS and exact solutions.

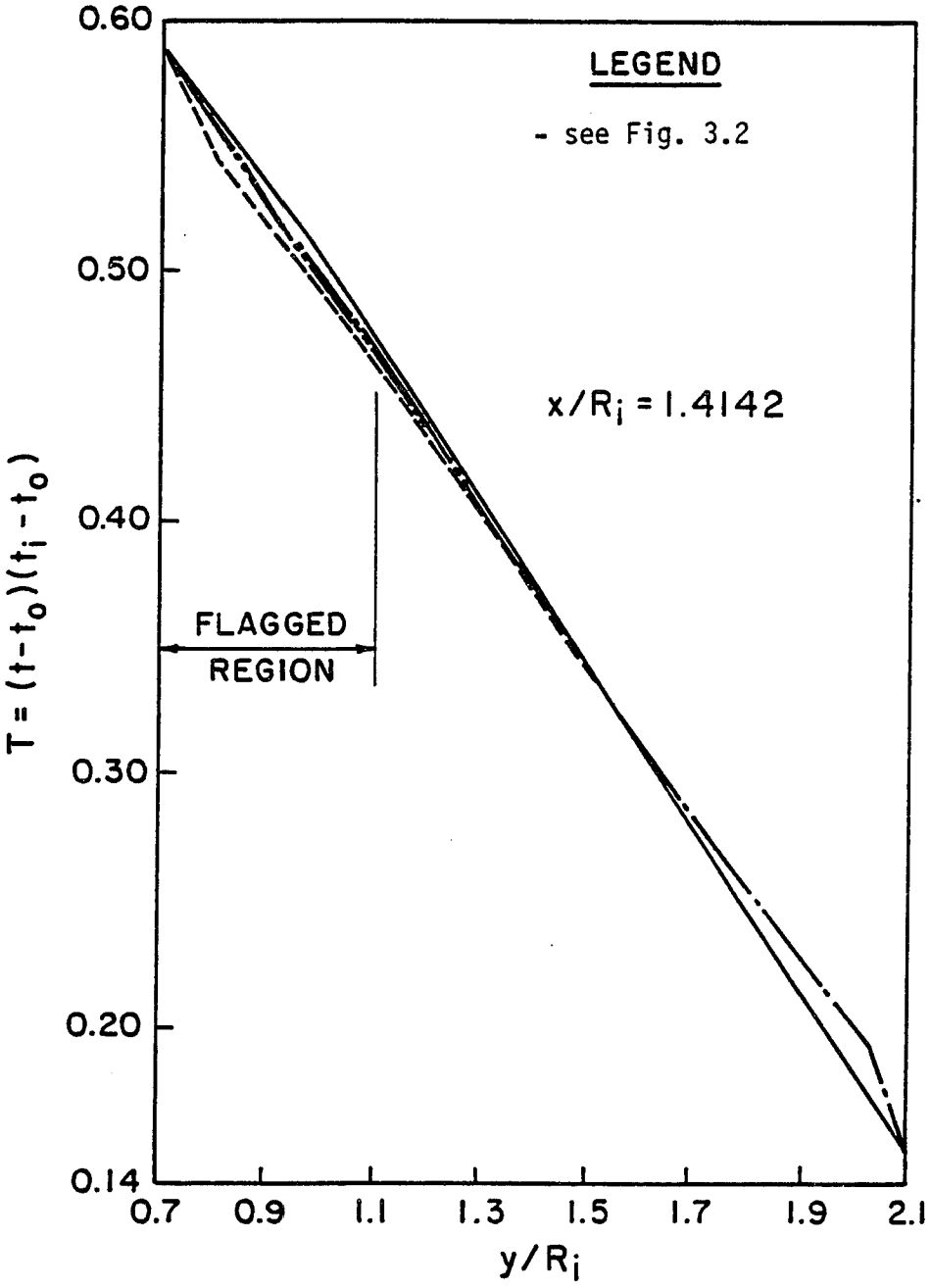


Fig. 3.4 Dimensionless temperature profile at $x/R_i = 1.4142$ for radial conduction in a rotating hollow cylinder. Comparing upwind, MAD1-WFDS and exact solutions.

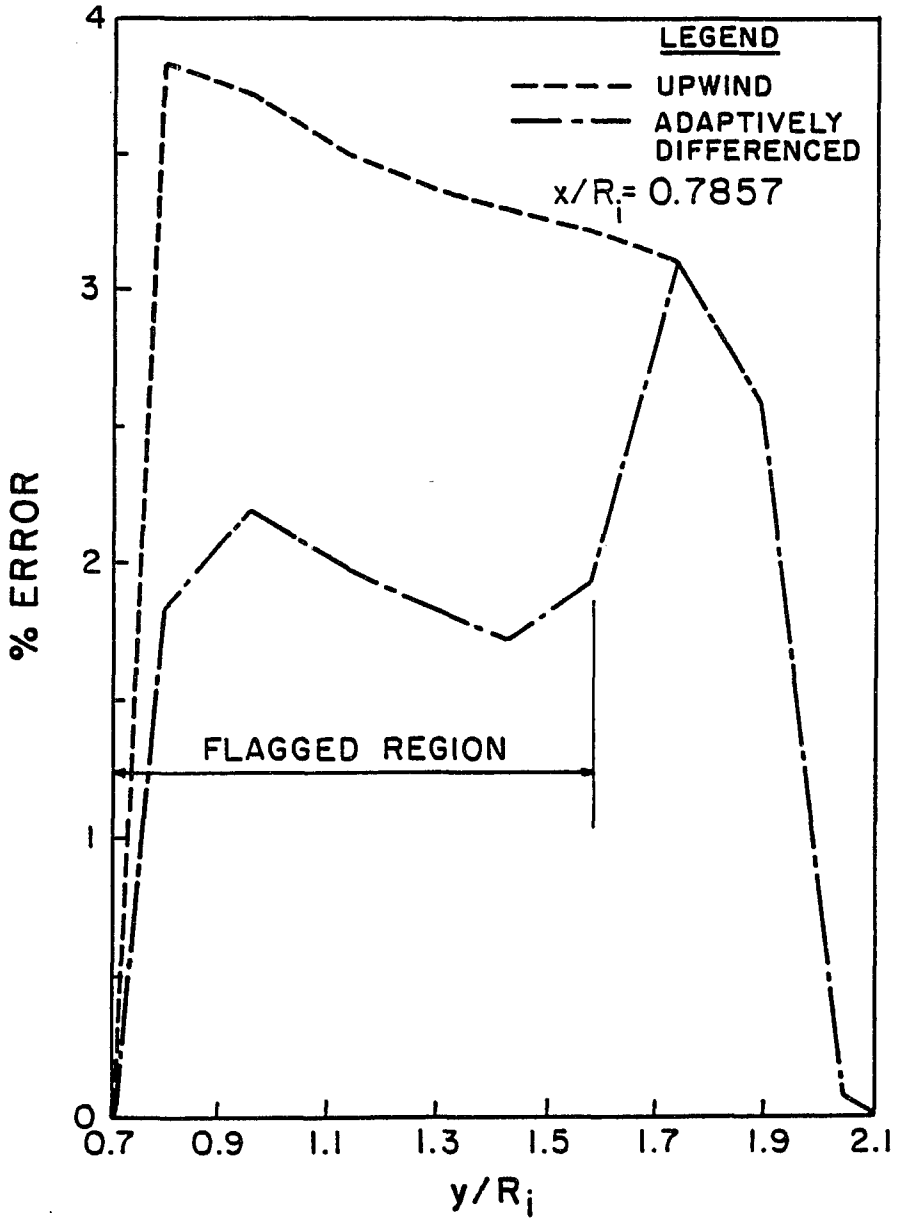


Fig. 3.5 Percent error at $x/R_i = 0.7857$ for radial conduction in a rotating hollow cylinder.

each plot, and the improvements to ϕ_0 outside the flagged regions, as affected by the multiple-grid strategy, are clearly evident in the figures. In Fig. 3.3, for example, the percentage error at $y/R_i = 1.5$ (outside the flagged region) for the upwind scheme is 3.3%, while the error is 2.2% for the MAD1-WFDS solution. Fig. 3.5 shows the percentage error at $x/R_i = 0.7857$. The adaptive differencing scheme clearly results in a smaller error than the upwind scheme in the flagged region. The maximum error within the flagged region for the MAD1-WFDS scheme is 2.2%, while for the upwind scheme the error is 3.9%.

3.2.2 Transport of a Step Change of a Scalar Variable

This problem was also presented in Chapter Two (see Section 2.7.2) and is not repeated here. The discretized domain and flagged region after the first level of flagging are shown in Fig. 3.6 for an 11 x 11 grid. As may be expected, the flagged region lies along the interface between ϕ=0 and ϕ=1.

Figures 3.7, 3.8 and 3.9 compare the results obtained using the upwind scheme and the adaptive method MAD1-WFDS with the exact solution for several values of x/L. The first stage upwind solution shows a considerable degree of smearing, due to false diffusion type errors. The second stage adaptively differenced solution shows considerable improvement, both in the flagged region Ω_{1i}, and in the unflagged region, $\Omega_0 - \Omega_{1i}$.

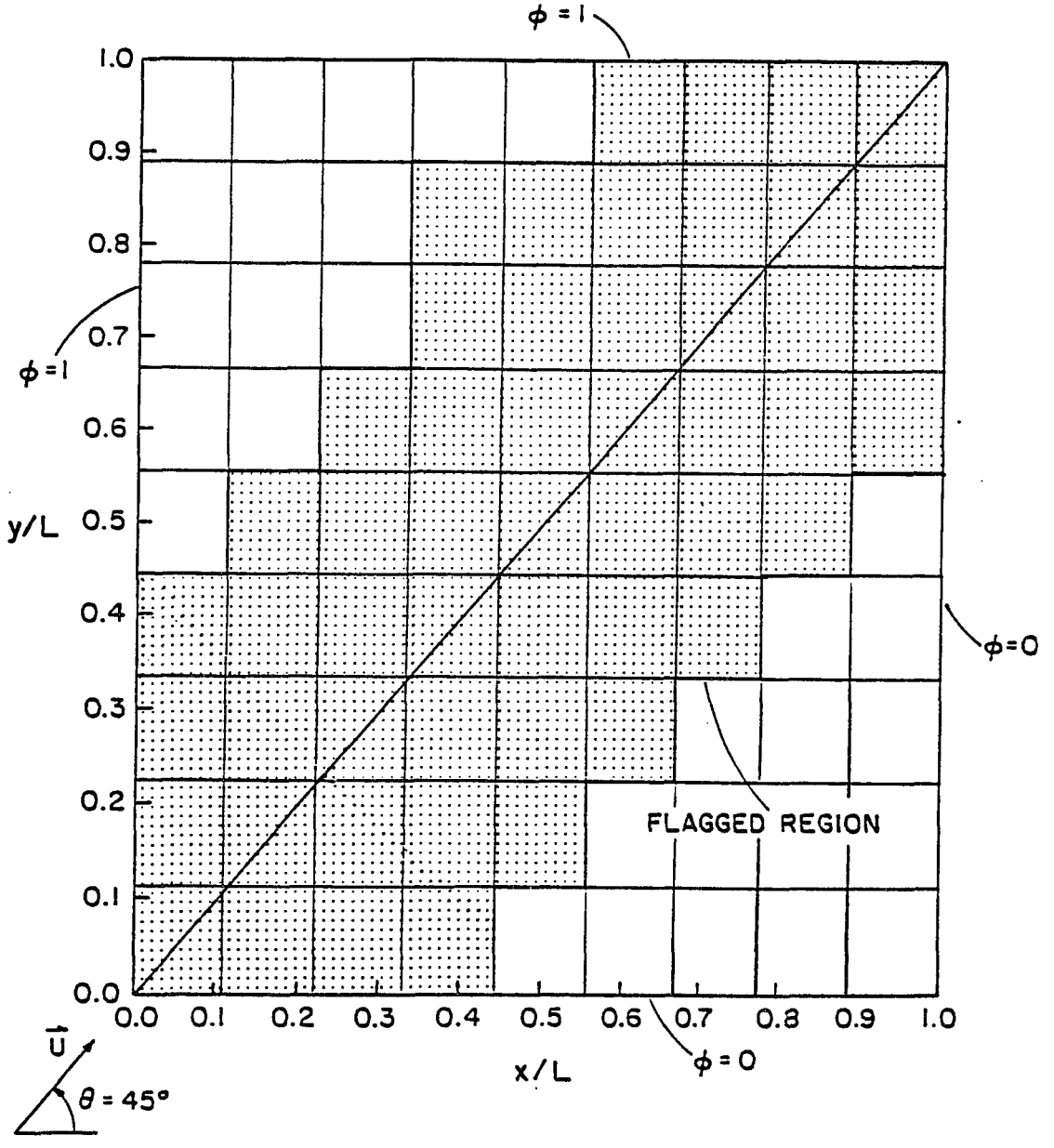


Fig. 3.6 Discretized physical domain and flagged region for the transport of a step change of a scalar variable in a region with a uniform velocity field.

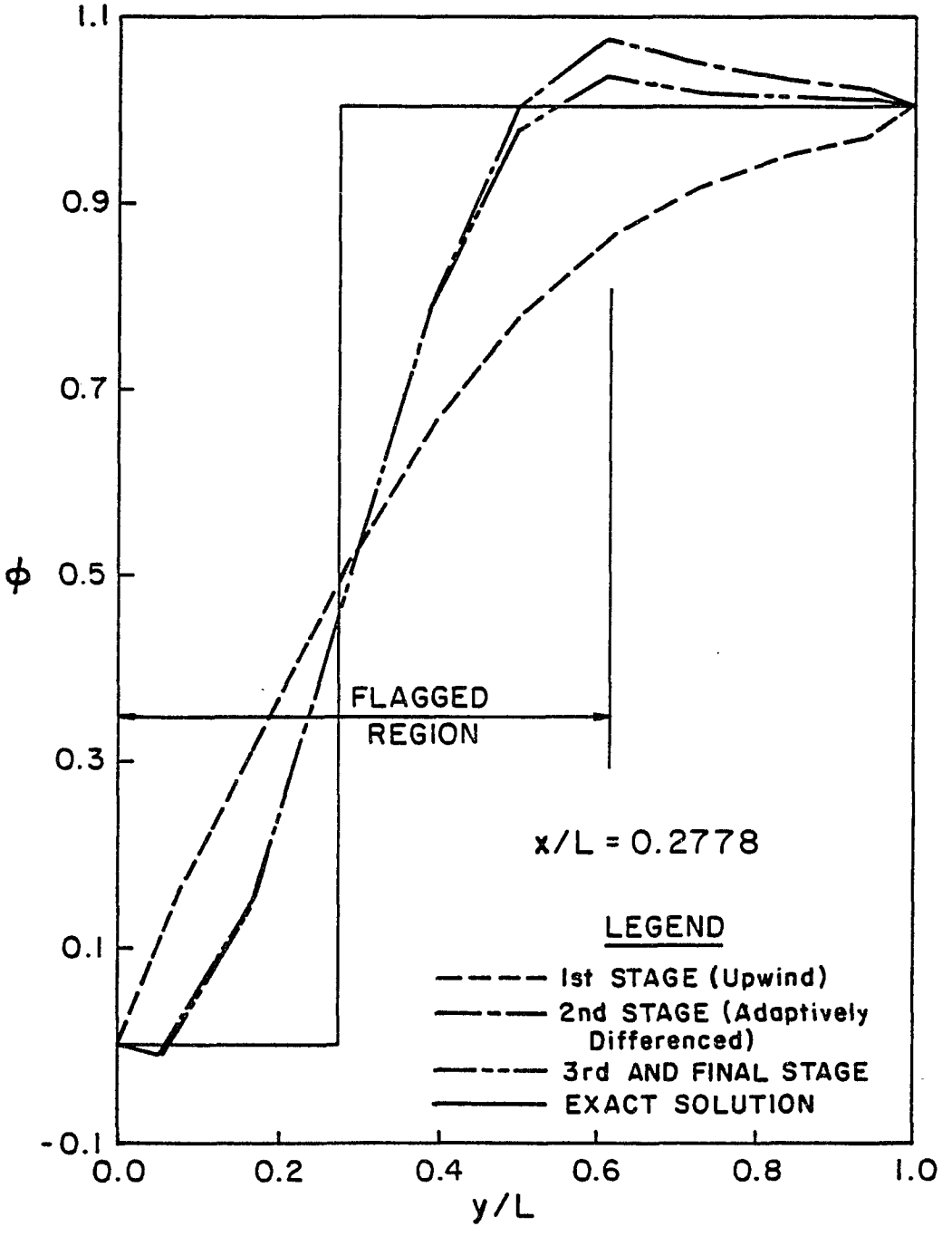


Fig. 3.7 ϕ profile at x/L = 0.2778 for the transport of a step change of a scalar variable in a region with a uniform velocity field. Comparing upwind, MAD1-WFDS and exact solutions.

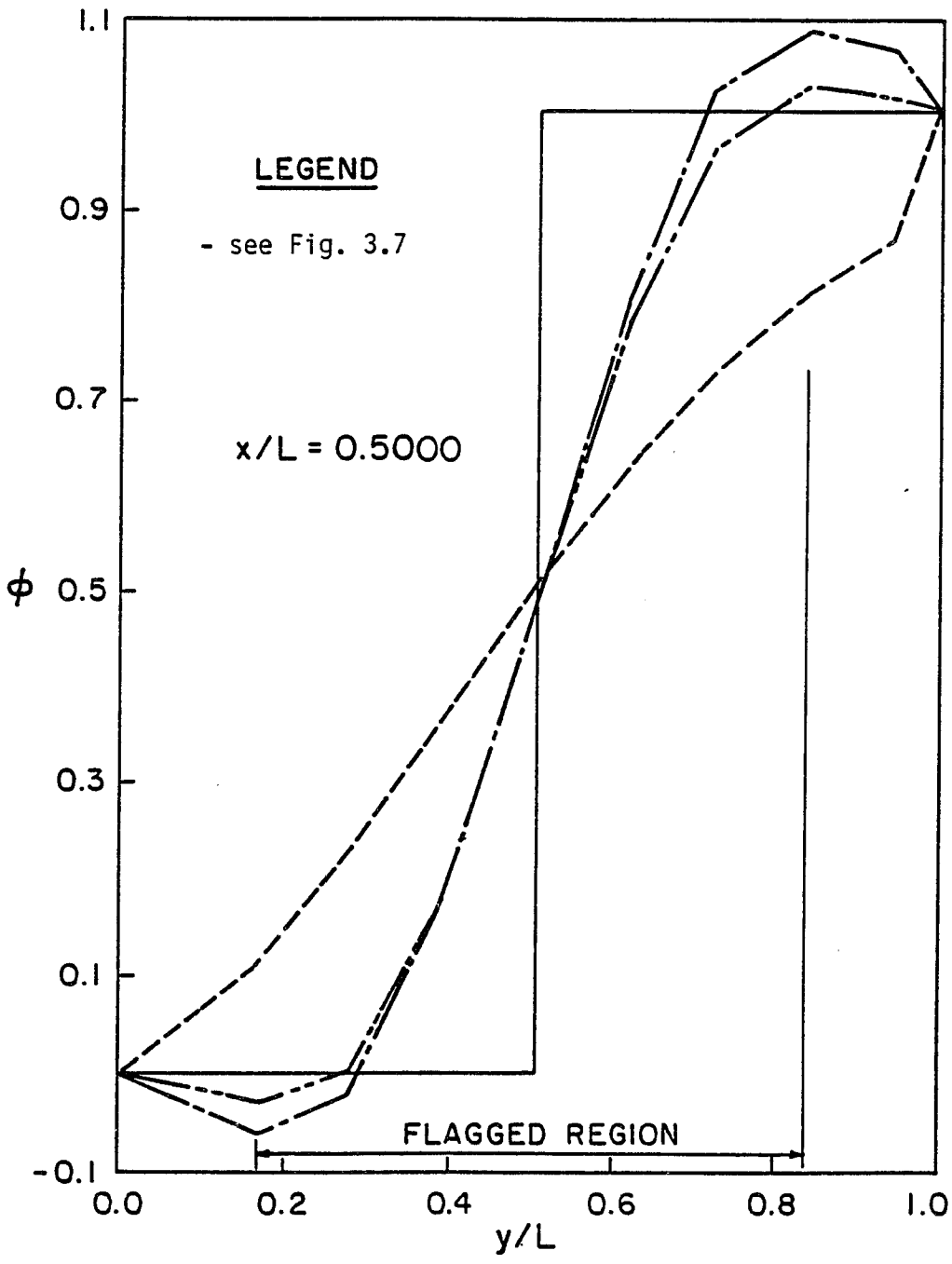


Fig. 3.8 ϕ profile at x/L = 0.5000 for the transport of a step change of a scalar variable in a region with a uniform velocity field. Comparing upwind, MAD1-WFDS and exact solutions.

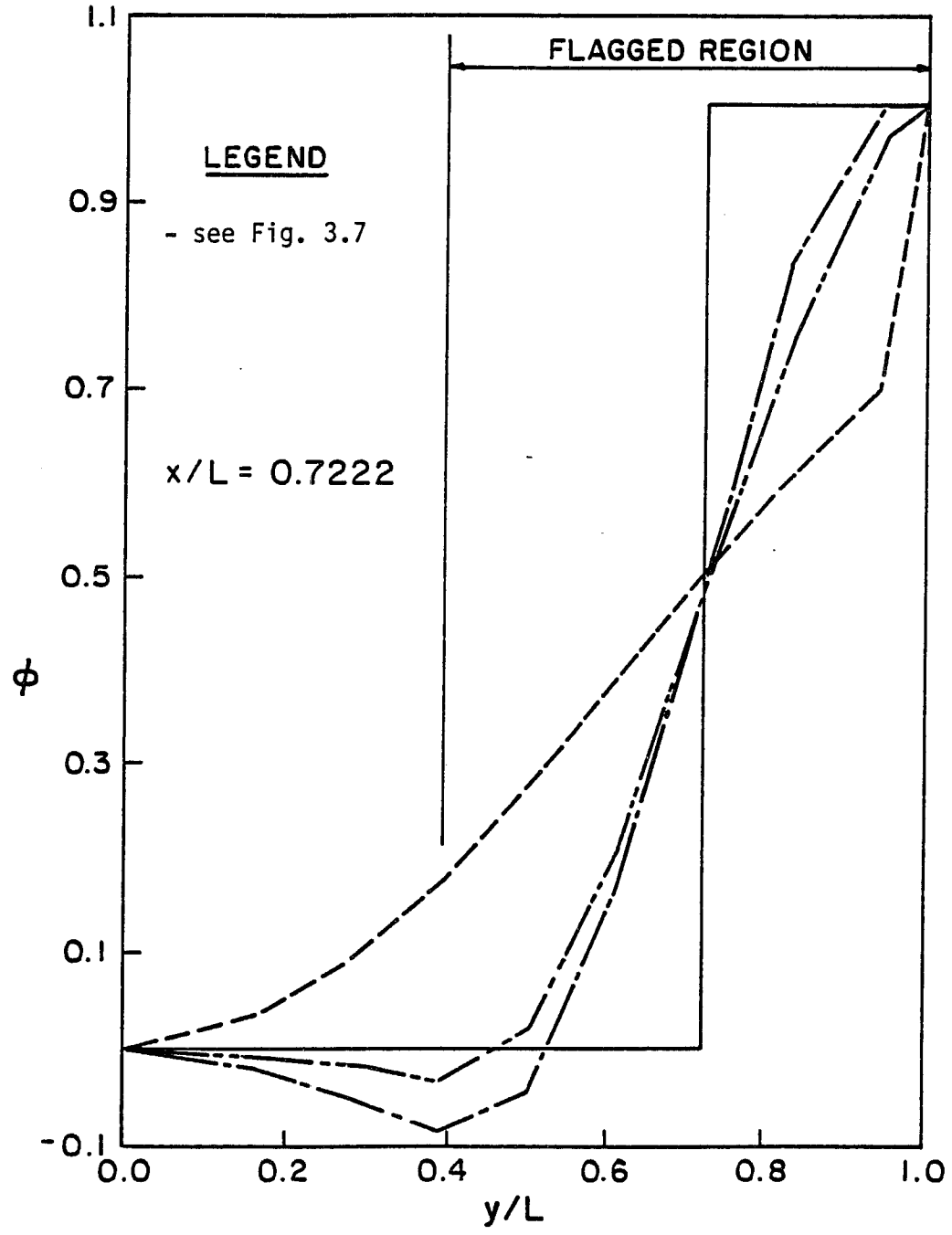


Fig. 3.9 ϕ profile at x/L = 0.7222 for the transport of a step change of a scalar variable in a region with a uniform velocity field. Comparing upwind, MAD1-WFDS and exact solutions.

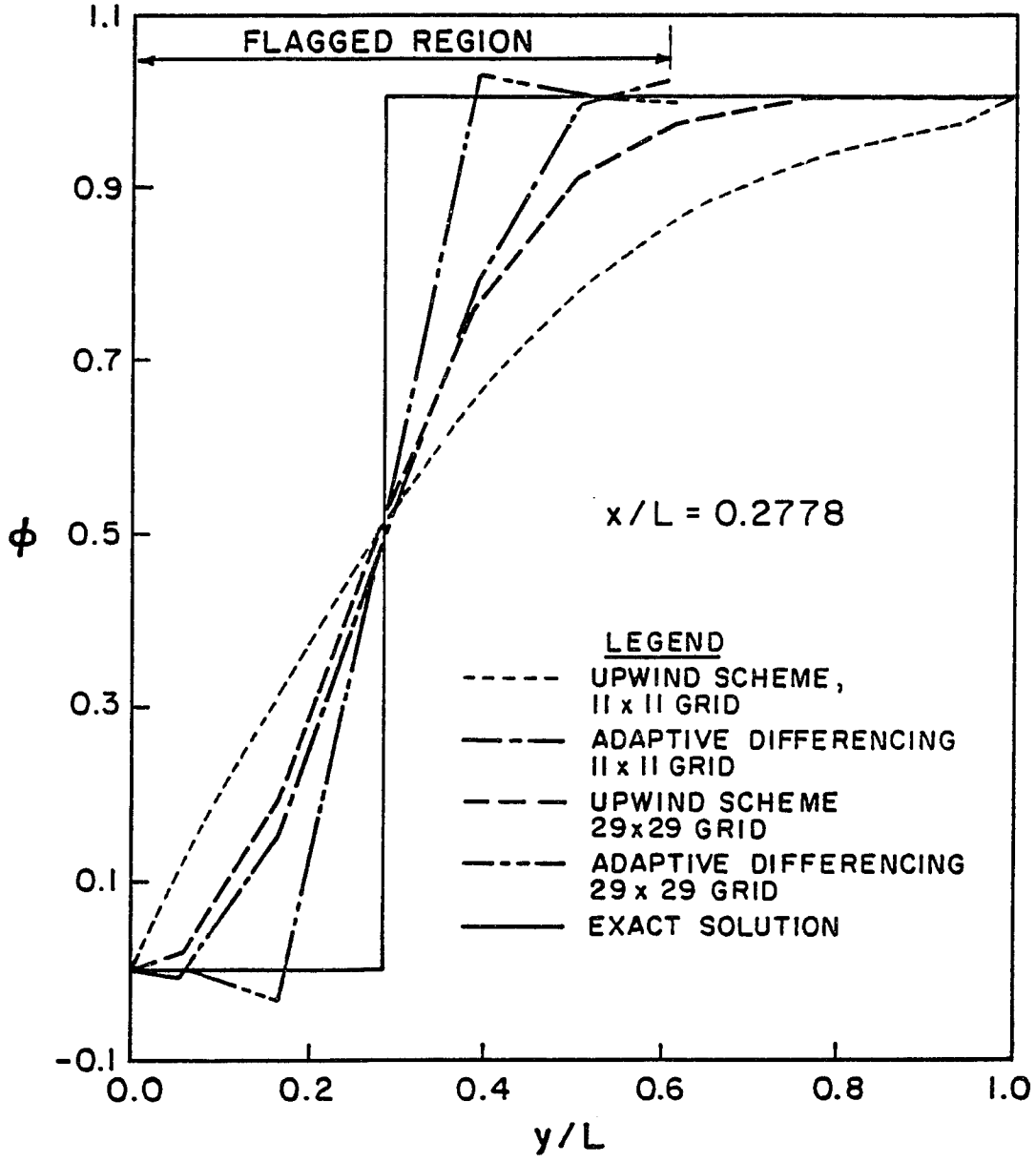


Fig. 3.10 ϕ profile at x/L = 0.2778 for the transport of a step change of a scalar variable in a region with a uniform velocity field. Comparing upwind and MAD1-WFDS solutions on 11x11 and 29x29 grids with exact solution.

The solution overshoots near y/L = 1.0 for x/L = 0.2778 and 0.5000, and undershoots for x/L = 0.722. These small overshoots and undershoots are typical of a higher order scheme. The third stage adaptively differenced solution improves the solution even further, and in particular, reduces the overshoots and undershoots considerably. The resulting third stage adaptively differenced solution is substantially better than the first stage upwind solution. For example, at y/L = 0.5 in Fig. 3.7 (x/L = 0.2778), the upwind solution percentage error is 27%, while the MAD1-WFDS solution error is 11%. At y/L = 0.72 in Fig. 3.8 (x/L = 0.50), the percentage errors are 25% and 3% for the upwind and MAD1-WFDS schemes, respectively. The improvements in the adaptively differenced solution, outside the flagged region, are quite clear and convincingly demonstrate the usefulness of a multigrid approach in the present adaptive differencing solution procedure. Comparing the solutions in Fig. 3.9 (x/L = 0.7222) at y/L = 0.3, the percentage errors are 10% and 2% for the upwind and MAD1-WFDS solutions, respectively.

Figure 3.10 shows the solutions on two different mesh sizes. The adaptively differenced solution is shown after the second stage, i.e., after a single application of Step 4 in the multigrid algorithm described earlier. The same conclusions hold for the finer 29 x 29 grid as described earlier for the 11 x 11 grid.

It should be noted that in making the previously described comparisons, no effort has been made to compare cpu times. A meaningful comparison would be to compare the times for the adaptively differenced solution with an upwind solution with the same level of accuracy. This would entail a number of trial upwind solutions on different grid sizes to determine which grid size results in the same error level as the adaptively differenced solution on a 11 x 11 grid. However, in Fig. 3.10, the 11 x 11 adaptively differenced solution is close to (and actually better than) the 29 x 29 upwind solution. Yet the cpu time for the adaptively differenced solution on the 11 x 11 mesh is only 1.26 seconds, while it is 6.26 seconds for the upwind solution on the 29 x 29 mesh (see Table 3.1). However, it should be pointed out that the use of the QUICK scheme not only improves the solution accuracy, but also reduces the number of iterations (and hence the cpu effort) needed to obtain a given degree of convergence. Therefore, a comparison of the cpu effort between schemes of different orders of accuracy that have differing convergence characteristics is somewhat ambiguous.

3.3 MULTIPLEGRID ADAPTIVE DIFFERENCING SCHEME 2 - WHOLE DOMAIN SWEEPS (MAD - WDS)

This algorithm employs the QUICK-based discretization scheme in the flagged regions and the upwind scheme in the

remainder of the problem domain. Unlike a true multigrid technique, this method solves the entire domain at each iteration using the following equations:

$$L_o \phi_o = b \quad in\ \Omega_o - \Omega_{1,i} \qquad 3.7a$$

$$L_1 \phi_1 = b \quad in\ \Omega_{1,i} \qquad 3.7b$$

Since the entire domain is solved at each iteration, the boundary values for the flagged regions are updated and improved at each iteration. Similarly, the solution at the unflagged points is improved since the solution at any point depends upon the neighboring values. The unflagged points bordering the flagged regions are improved at each iteration, and, in turn, feed improved values further into the unflagged domain.

Algorithm Two may be stated as follows:

1. In Ω_0, obtain the first order solution $L_0\phi_0 = b$.
2. Flag grid points with error estimates E_n greater than a specified tolerance, and define $\Omega_{1,i}$.
3. Solve

   $L_0\phi_0 = b \quad$ in $(\Omega_0 - \Omega_{1,i})$

   $L_1\phi_1 = b \quad$ in $\Omega_{1,i}$

   in an iterative manner until the desired accuracy level is satisfied.
4. Proceed to the next level of adaptive differencing by defining $\Omega_{2,i}$, which will generally be nested in $\Omega_{1,i}$.

5. Solve

$$L_0\phi_0 = b \quad \text{in } (\Omega_0 - \Omega_{1,i} - \Omega_{2,i})$$

$$L_1\phi_1 = b \quad \text{in } (\Omega_0 - \Omega_{1,i})$$

$$L_2\phi_2 = b \quad \text{in } \Omega_{2,i}$$

until the desired accuracy level is achieved.

This algorithm (MAD2-WDS) is applied to the same two convection-diffusion problems.

3.3.1 Radial Heat Conduction in a Rotating Hollow Cylinder

Again, the numerical results were computed on an 11 x 11 grid. Figure 3.11 is a plot of the dimensionless temperature profile as a function of y/R_i for the rotating cylinder problem. The same three x/R_i locations plotted for multigrid adaptive differencing scheme one (MAD1-WFDS) are used in multiple-grid method two (MAD2-WDS). The multiple-grid method is compared with the upwind and exact solutions. The adaptively differenced solution is clearly more accurate than the upwind solution at all three x/R_i locations. The maximum percentage errors for the upwind solution (within the flagged region) are 3.83%, 3.51%, and 3.06% at x/R_i = 0.7857, 1.0999, and 1.4142, respectively. The maximum percentage errors for the MAD2-WDS solution (within the flagged region) are 2.20%, 2.27%, 2.31% at x/R_i = 0.7857, 1.0999, and 1.4142, respectively.

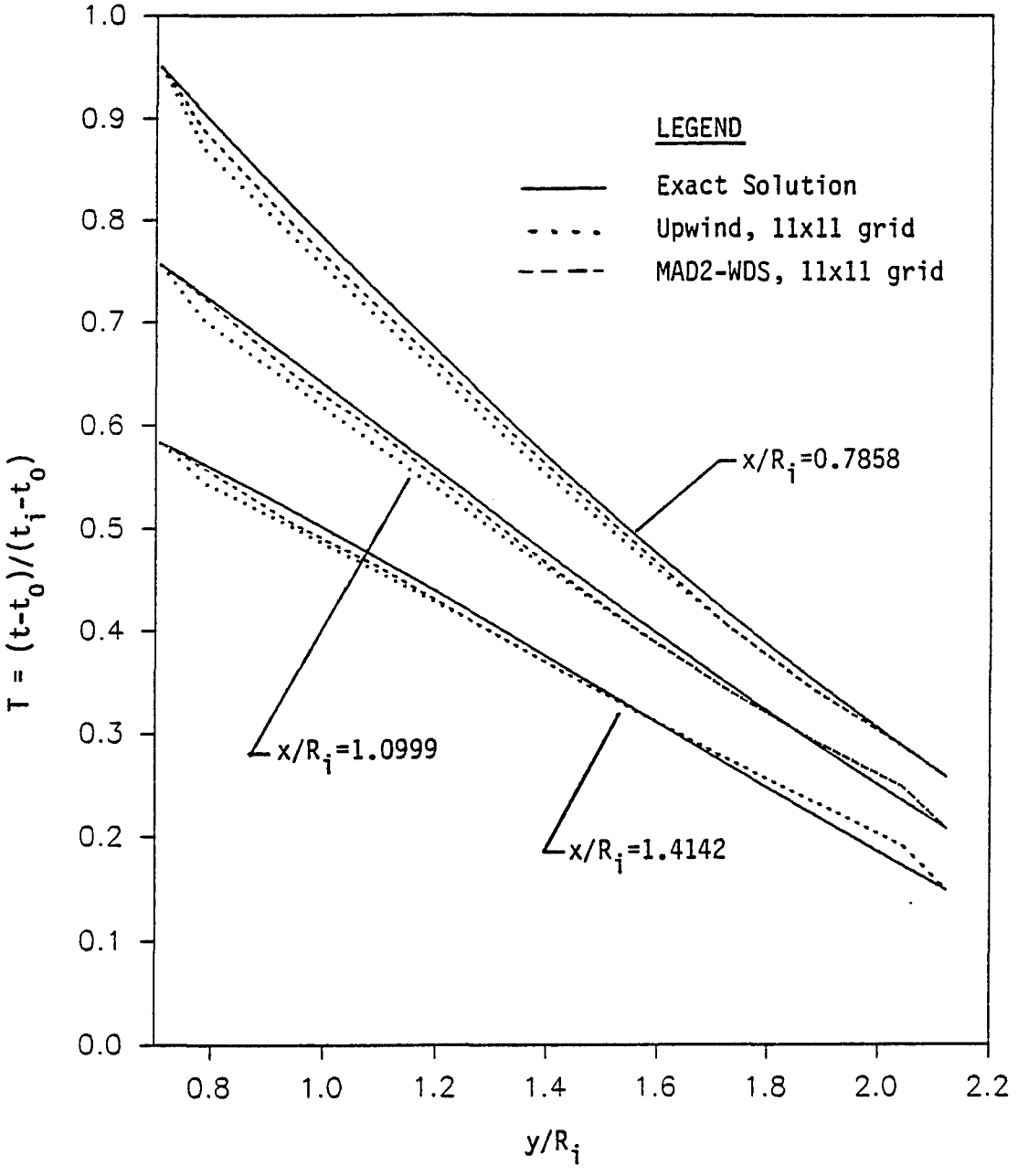


Fig. 3.11 Dimensionless temperature profile at x/R_i = 0.7857, 1.0999, and 1.1412 for radial conduction in a rotating hollow cylinder. Comparing upwind, MAD2-WDS and exact solutions.

3.3.2 Transport of a Step Change of a Scalar Variable

Figures 3.12, 3.13 and 3.14 compare the results obtained using the upwind scheme and MAD2-WDS with the exact solution for three values of x/L for the step change of a variable problem on an 11 x 11 mesh. Again, this MAD2-WDS method shows an improvement over the upwind method. Note that the MAD2-WDS solution primarily overlays the QUICK method solution in the flagged regions. In Fig. 3.12 (x/L = 0.2778), the upwind scheme yields a 22% error at y/L = 0.50, while the MAD2-WDS scheme results in only a 2% error at the same y/L location. The percentage errors at y/L = 0.4 in Fig. 3.13 (x/L = 0.50) are 38% and 22% for the upwind and MAD2-WDS solutions, respectively.

3.4 MULTIPLEGRID ADAPTIVE DIFFERENCING SCHEME 3 - FLAGGED DOMAIN SWEEPS (MAD3 - FDS)

The third multigrid method is similar to Multigrid Method No. 1. In this procedure, however, after the solution is obtained on the flagged region(s), a solution on the unflagged regions rather than on the entire global domain is then sought. The interior boundaries of the unflagged regions overlap the flagged boundaries by one grid point so that the newly-generated improved QUICK solution obtained in the flagged subdomains is used as the boundary conditions for the unflagged regions. The algorithm for this method is as follows:

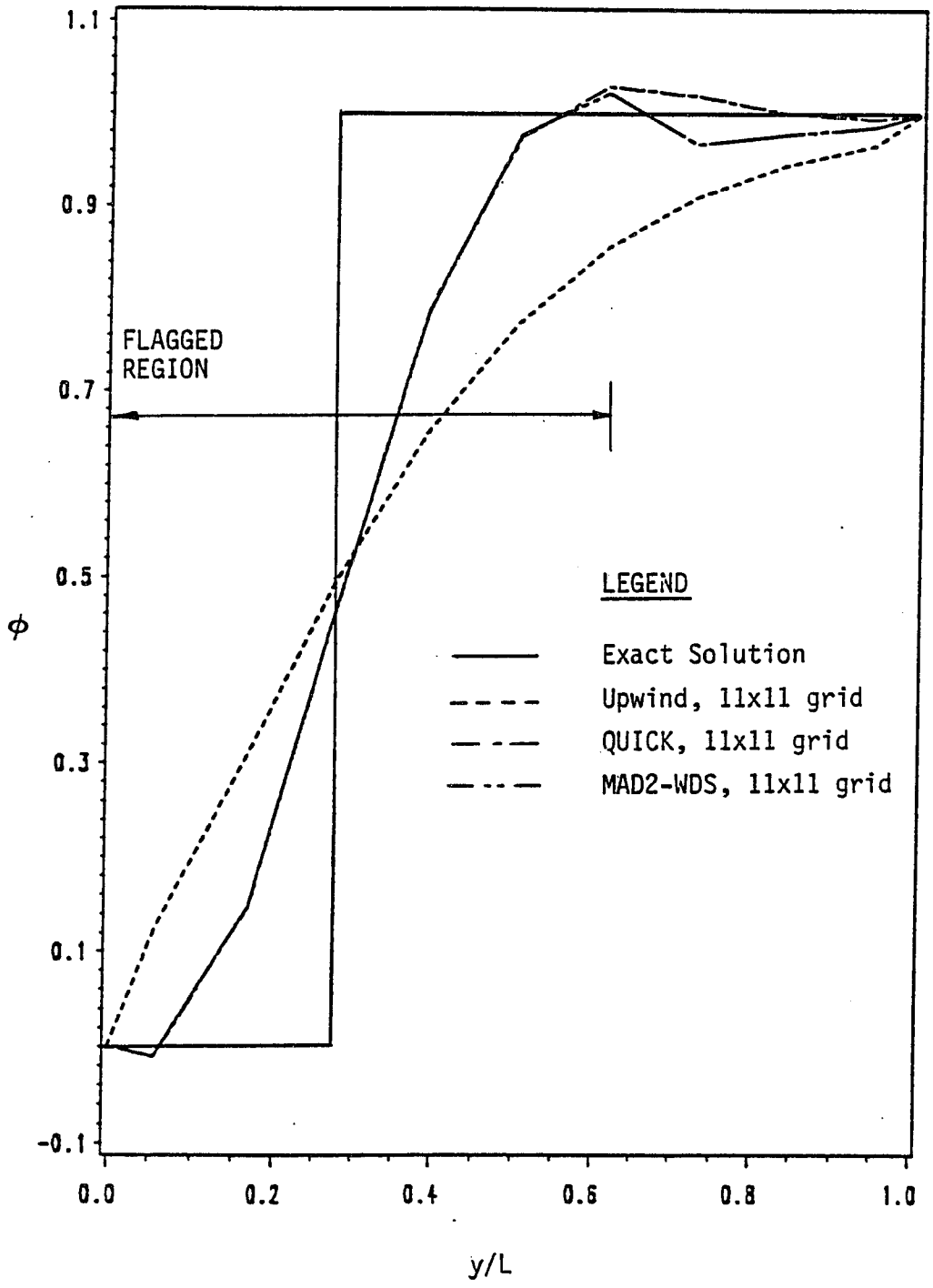


Fig. 3.12 ϕ profile at x/L = 0.2778 for the transport of a step change of a scalar variable in a region with a uniform velocity field. Comparing upwind, QUICK, MAD2-WDS and exact solutions.

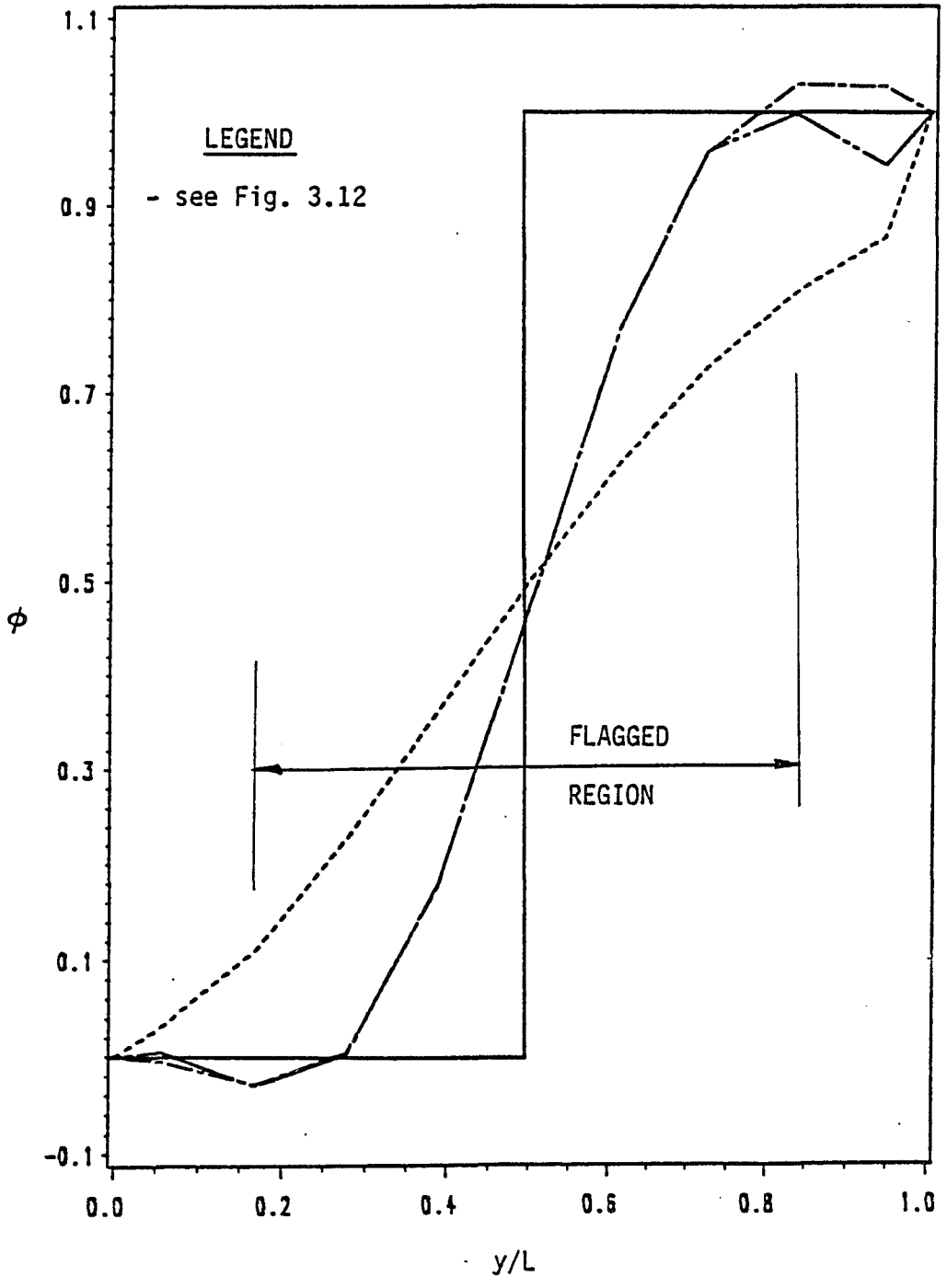


Fig. 3.13 ϕ profile at x/L = 0.5000 for the transport of a step change of a scalar variable in a region with a uniform velocity field. Comparing upwind, QUICK, MAD2-WDS and exact solutions.

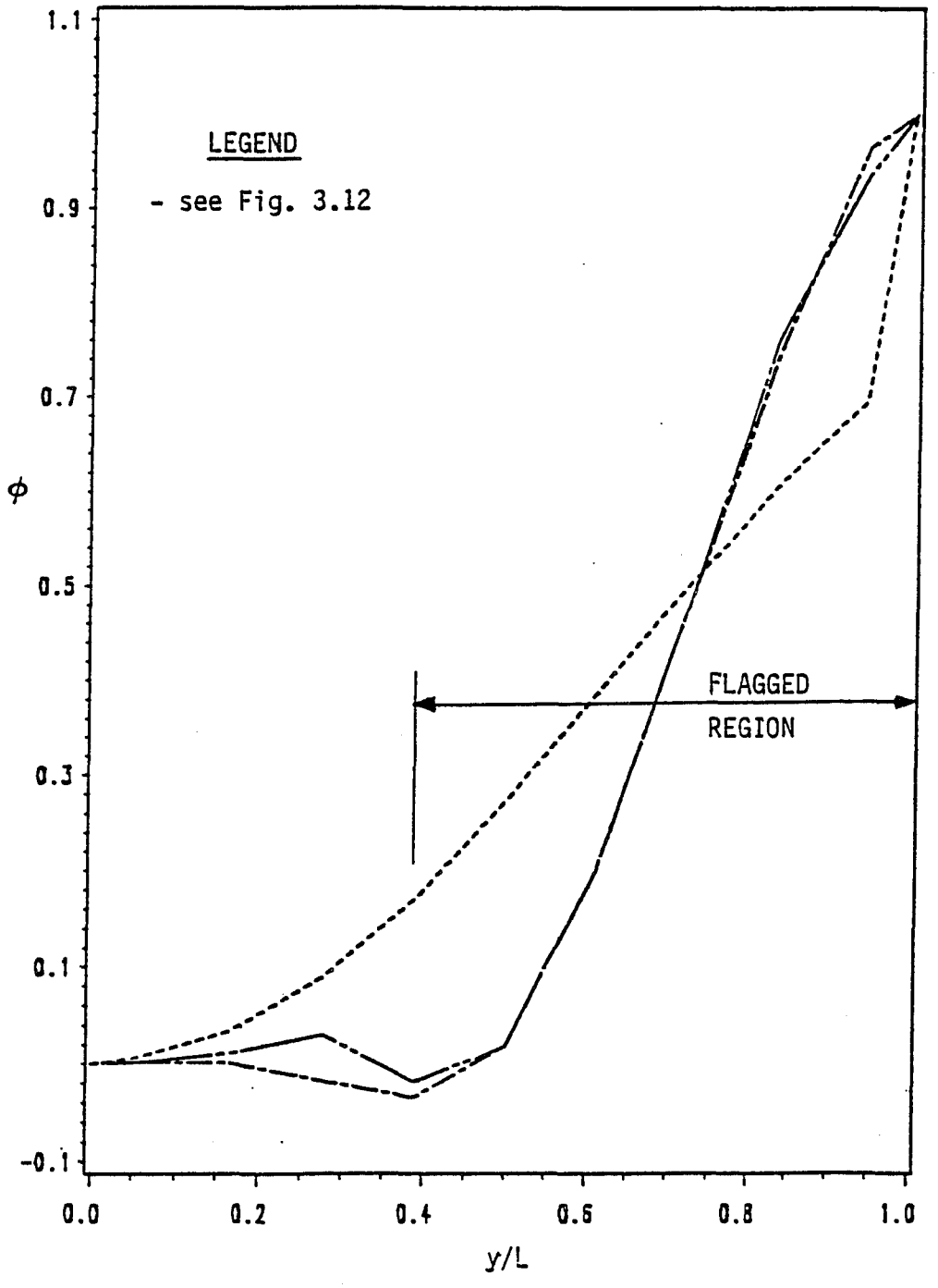


Fig. 3.14 ϕ profile at x/L = 0.7222 for the transport of a step change of a scalar variable in a region with a uniform velocity field. Comparing upwind, QUICK, MAD2-WDS and exact solutions.

1. In Ω_0, obtain the first order solution $L_0\phi_0 = b$.
2. Flag grid points with error estimates E_n greater than a specified tolerance, and define $\Omega_{1,i}$.
3. In $\Omega_{1,i}$, obtain the third order, QUICK solution $L_1\phi_1 = b$ with boundary conditions based on ϕ_0.
4. In $(\Omega_0 - \Omega_{1,i})$, obtain improved solutions by solving $L_0\phi_0 = b$ in $(\Omega_0 - \Omega_{1,i})$ with boundary conditions based on $\phi_{1,i}$.
5. Return to Step 3 and repeat until the desired accuracy level is satisfied.
6. Proceed to the next level of adaptive differencing by defining $\Omega_{2,i}$, which will generally be nested in $\Omega_{1,i}$, and repeat above steps. Continue to the desired accuracy levels.

MAD3-FDS is applied to the rotating hollow cylinder and transport of a step change of a scalar variable problems. The results are compared with the exact and upwind solutions.

3.4.1 Radial Heat Conduction in a Rotating Hollow Cylinder

Figure 3.15 is a plot of the dimensionless temperature profile as a function of y/R_i for the rotating cylinder problem. The same three x/R_i locations plotted for

the two previous multigrid methods are used to show MAD3-FDS. Again, the multigrid method is compared with the upwind and exact solutions on an 11 x 11 mesh. Like MAD1-WFDS and MAD2-WDS, MAD3-FDS yields results closer to the exact solution than the upwind scheme. The maximum percentage error within the flagged region at x/R_i = 0.7857 are 3.8% and 2.19% for the upwind and MAD3-FDS solutions, respectively. At x/R_i = 1.0999, 3.51% and 2.06% are the maximum errors for the upwind and MAD3-FDS solutions, respectively. At x/R_i = 1.4142, the maximum percentage errors (within the flagged region) are 3.15% for the upwind scheme and 2.28% for the MAD3-FDS solution.

Again, note the improvements to the MAD3-FDS solution outside of the flagged region. Examining the solution at the second grid point outside the flagged region at each x/R_i location, the percentage errors for the upwind solution are 2.58% at x/R_i = 0.7857, 1.66% at x/R_i = 1.0999 and 1.54% at x/R_i = 1.4142. The errors for the MAD3-FDS solution (again, at the second grid point outside of the flagged region) are 2.57%, 1.63% and 1.02% at x/R_i = 0.7857, 1.0999, and 1.4142, respectively.

3.4.2 Transport of a Step Change of a Scalar Variable

Figures 3.16, 3.17 and 3.18 compare the results obtained using the upwind scheme and MAD3-FDS with the exact solution for three values of x/L for the step change of a variable

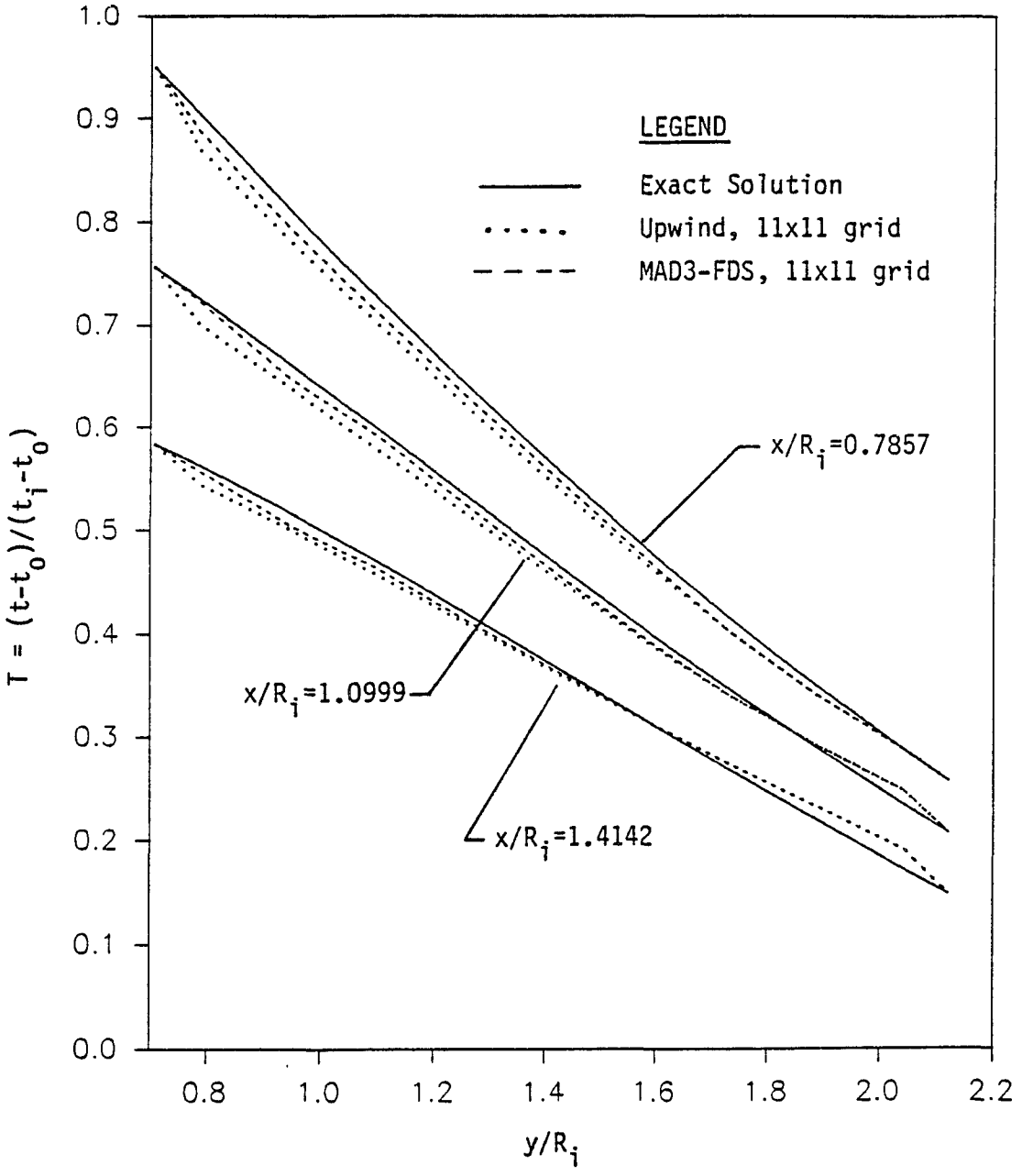


Fig. 3.15 Dimensionless temperature profile at x/R_i = 0.7857, 1.0999 and 1.4142 for radial conduction in a rotating hollow cylinder. Comparing upwind, MAD3-FDS and exact solutions.

problem. This time, the results are shown for a 73 x 73 grid. Each plot presents the upwind, QUICK, MAD3-FDS and exact solutions. The QUICK and MAD3-FDS solutions overlay one another within the flagged region. The multigrid scheme clearly results in an improved solution over the upwind solution, both in and outside of the flagged region. At x/L = 0.2778 (see Fig. 3.16), the upwind solution is ϕ = 0.32 inside the flagged region at y/L = 0.3, the corresponding MAD3-FDS solution is ϕ = -0.01, while the exact solution is ϕ = 0. Outside the flagged region (in Fig. 3.16), at y/L = 0.6, the upwind, MAD3-FDS and exact solutions are ϕ = 0.98, 0.998, and 1.0, respectively.

3.5 COMPARISON OF THE MULTIGRID METHODS

The three multiple grid adaptive differencing (MAD) schemes result in very similar solutions as may be seen by examining Figs. 3.19 through 3.22. Figure 3.19 shows the temperature plots for the rotating cylinder problem on a 11 x 11 grid. The three MAD solutions overlay one another on this scale. Figures 3.20, 3.21 and 3.22 are plots of the percent error of the three schemes for this problem. Again, they demonstrate the similar solutions generated by the three methods. The differences in the percentage errors for the three MAD schemes may be compared by computing the following,

$$| \text{Scheme A} - \text{Scheme B} | \times 100$$

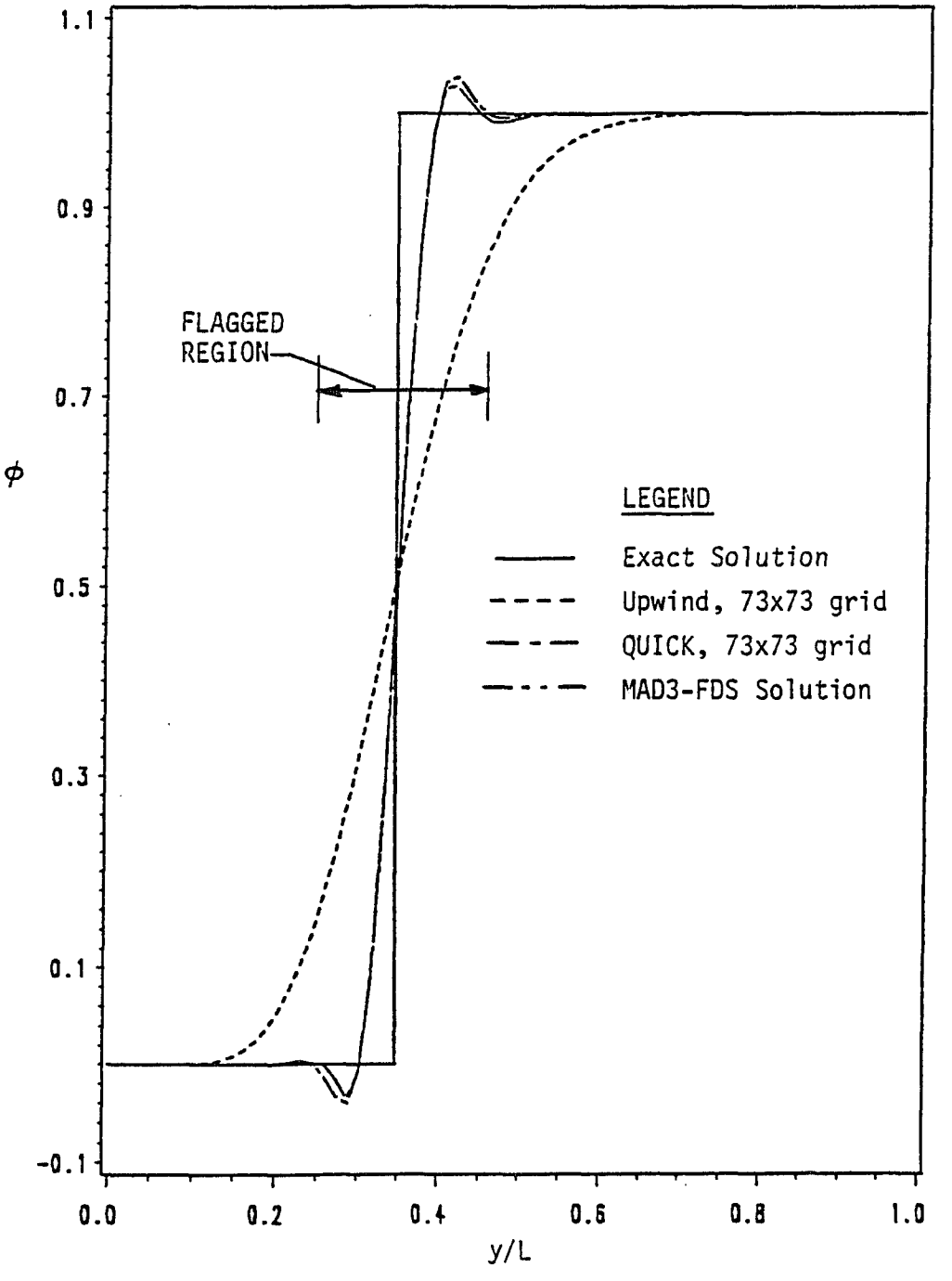


Fig. 3.16 ϕ profile at x/L = 0.2778 for the transport of a step change of a scalar variable in a region with a uniform velocity field. Comparing upwind, MAD3-FDS and exact solutions

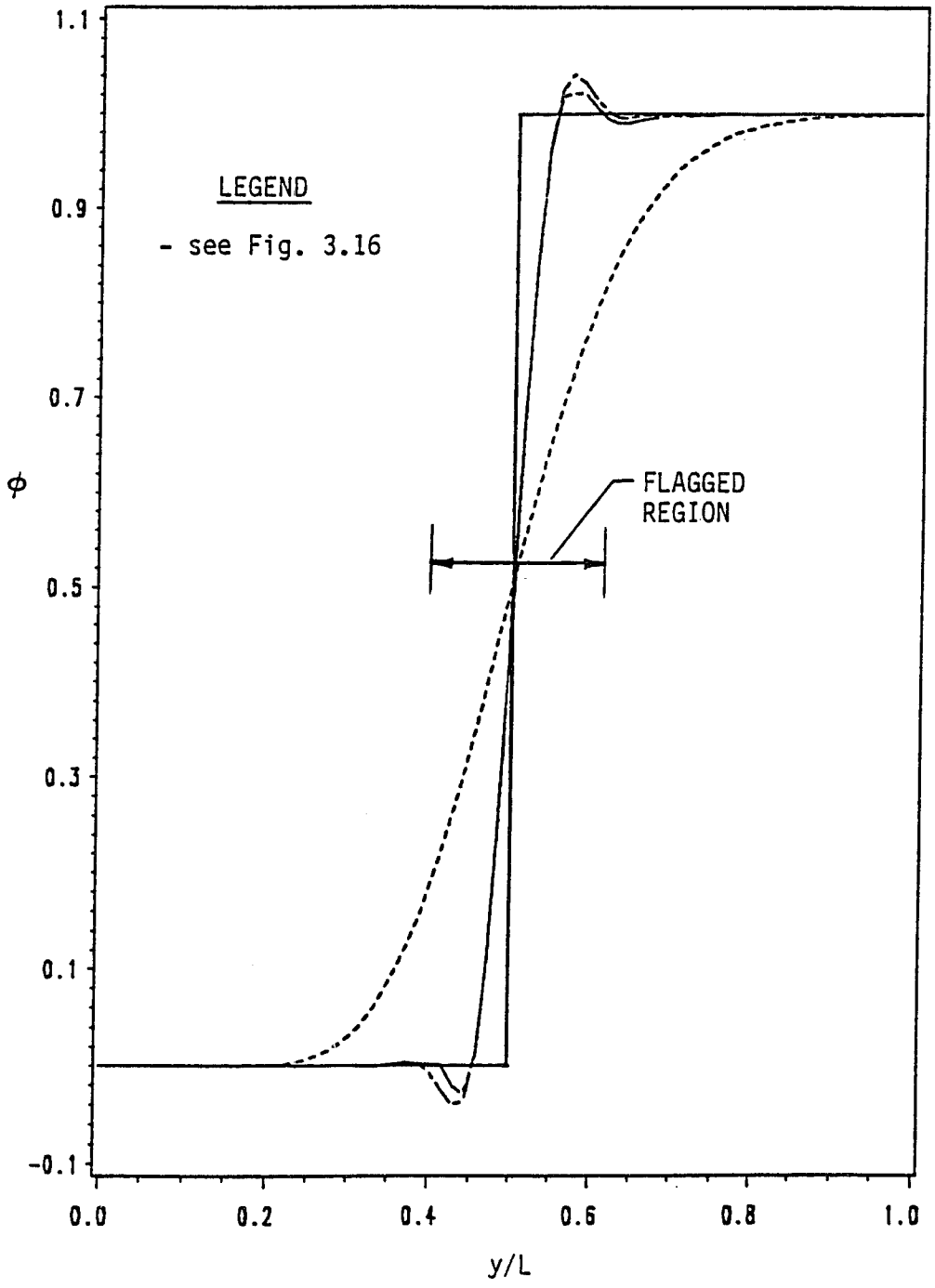


Fig. 3.17 ϕ profile at x/L = 0.5000 for the transport of a step change of a scalar variable in a region with a uniform velocity field. Comparing upwind, MAD3-FDS and exact solutions

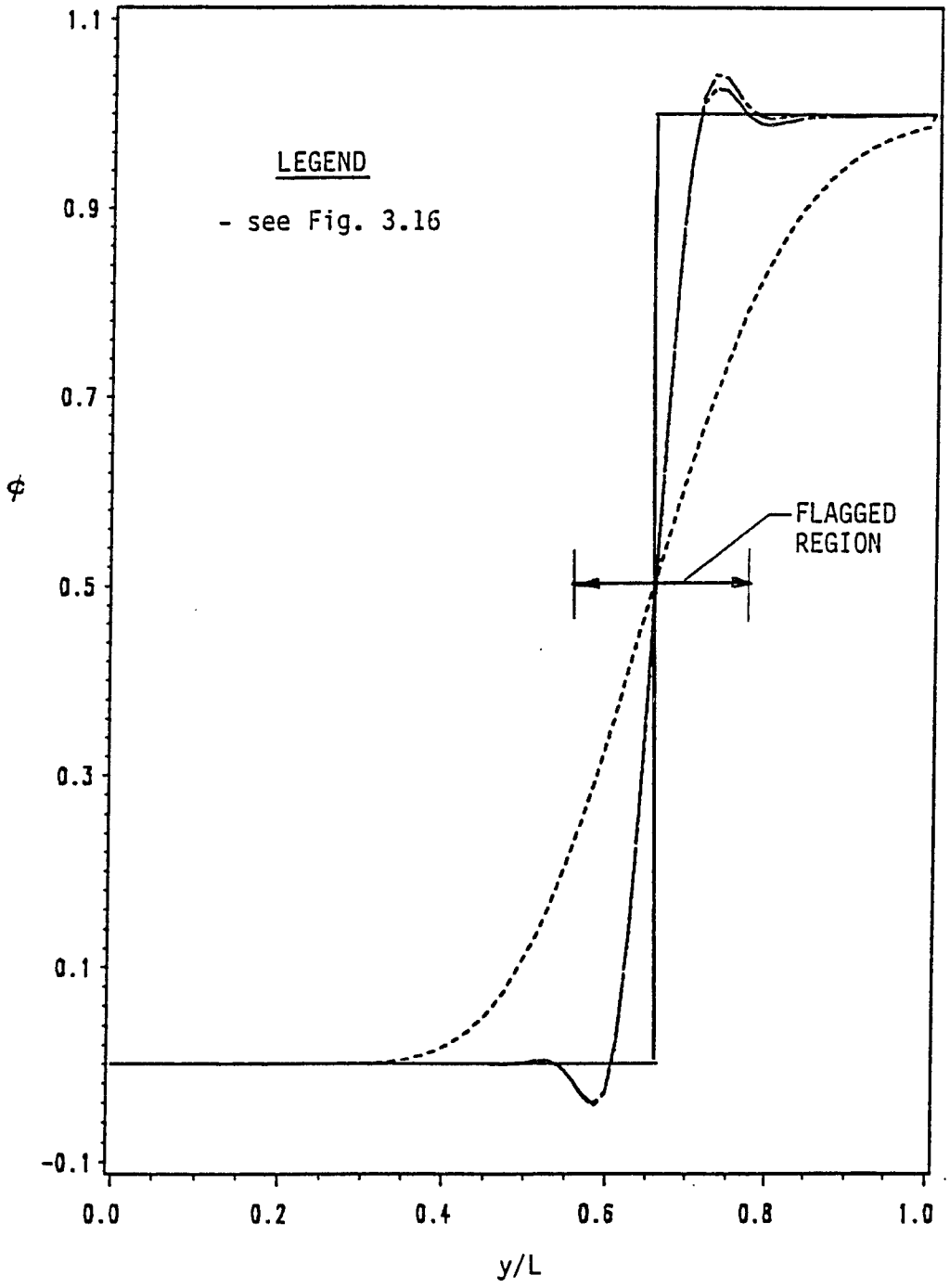


Fig. 3.18 ϕ profile at x/L = 0.7222 for the transport of a step change of a scalar variable in a region with a uniform velocity field. Comparing upwind, MAD3-FDS and exact solutions.

where Scheme A and Scheme B may be any two of the three MAD methods. At $x/R_i = 0.7857$, the maximum difference in percentage error between the three methods is only 0.35%. It is 0.37% and 0.4% at $x/R_i = 1.0999$ and 1.4142, respectively.

Next, the three multigrid solutions for the problem of the transport of a step change of a scalar variable are overlayed on the same axes and compared with the exact solution for a 73 x 73 grid in Figs. 3.23 through 3.25. Here, also, the adaptive differencing solutions are so similar that they overlay one another.

Table 3.1 and Table 3.2 show the computer timing information for both test case problems. Both virtual and total cpu times are presented for comparison. As mentioned earlier, the QUICK scheme (for the 11 x 11 grid) requires the least cpu time, since the solution scheme requires fewer iterations to converge than the upwind scheme for these convection-diffusion problems. The MAD schemes all require more cpu effort than the upwind scheme for the same mesh size. However, less time is required than for the 29 x 29 upwind solution (which is still not as accurate as the MAD schemes).

For the step change of a scalar variable problem, the upwind, QUICK and MAD1-WFDS schemes were run on a 73 x 73 mesh. On this larger grid, the QUICK solution scheme requires more computer time than the upwind method. Here the advantage of the MAD schemes in computational savings is evident. This

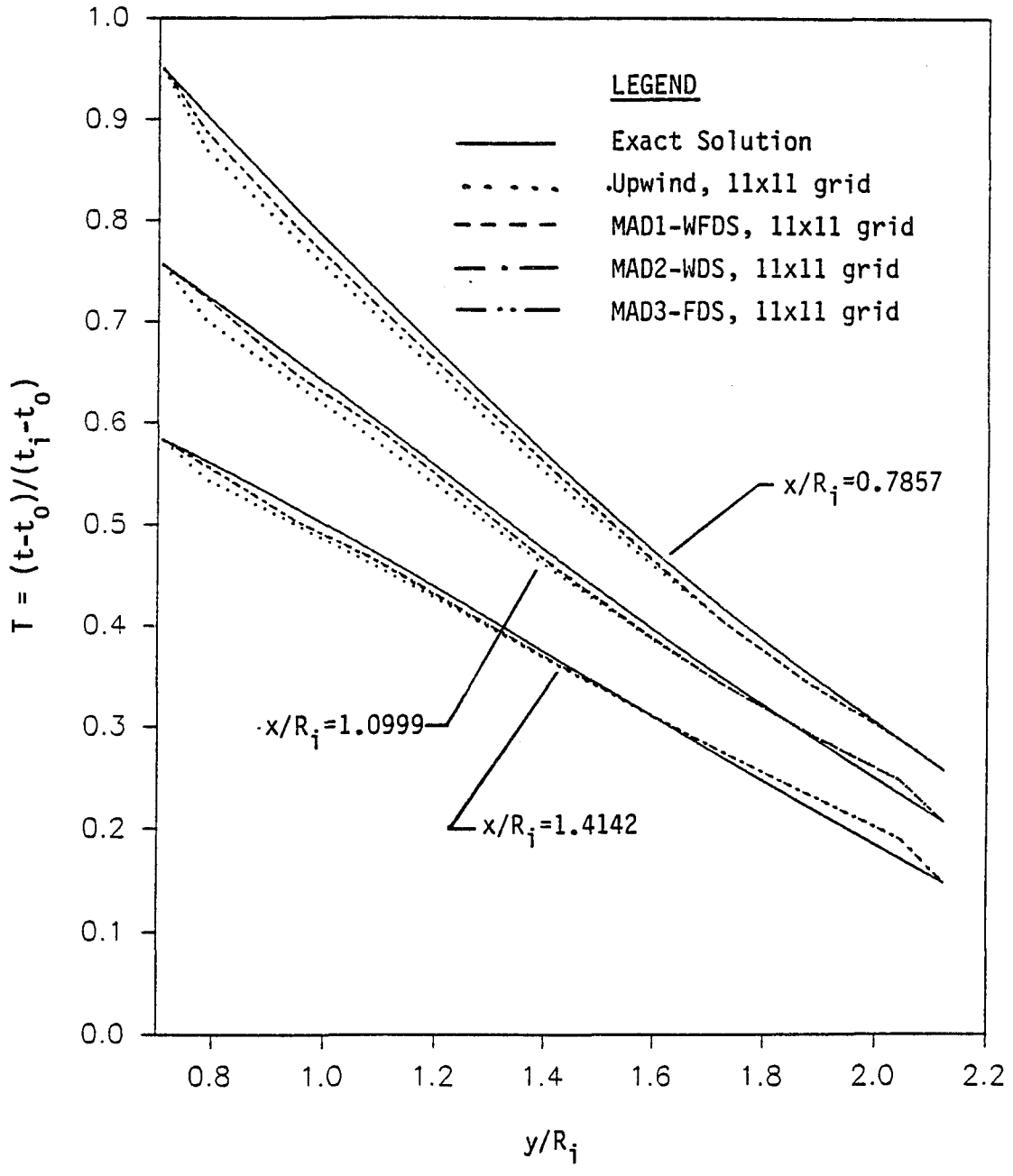


Fig. 3.19 Dimensionless temperature profile at x/R_i = 0.7857, 1.0999 and 1.4142 for radial conduction in a rotating hollow cylinder. Comparing MAD1-WFDS, MAD2-WDS, MAD3-FDS and exact solutions.

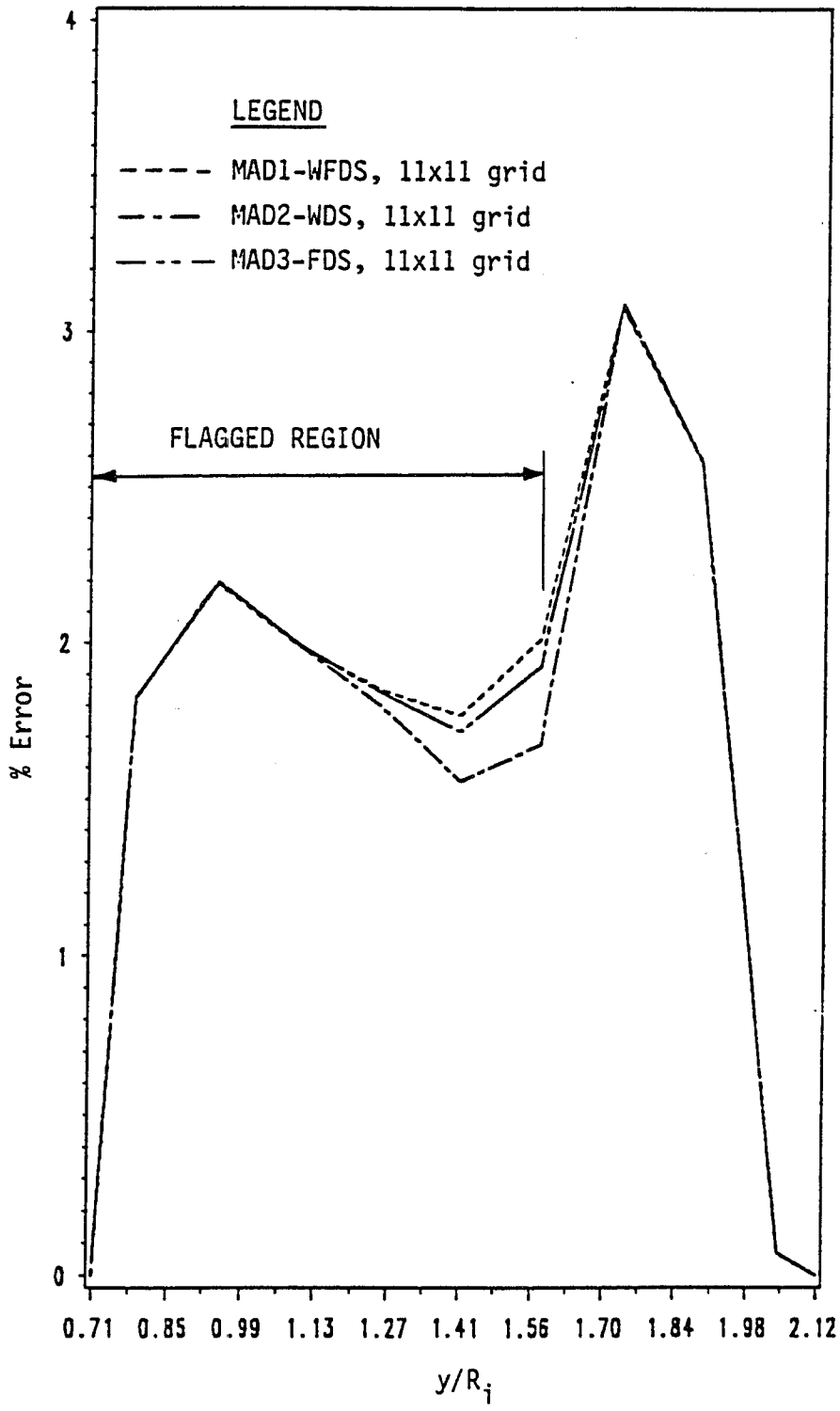


Fig. 3.20 Percent error at $x/R_i = 0.7857$ for radial conduction in a rotating hollow cylinder. Comparing MAD1-WFDS, MAD2-WDS and MAD3-FDS.

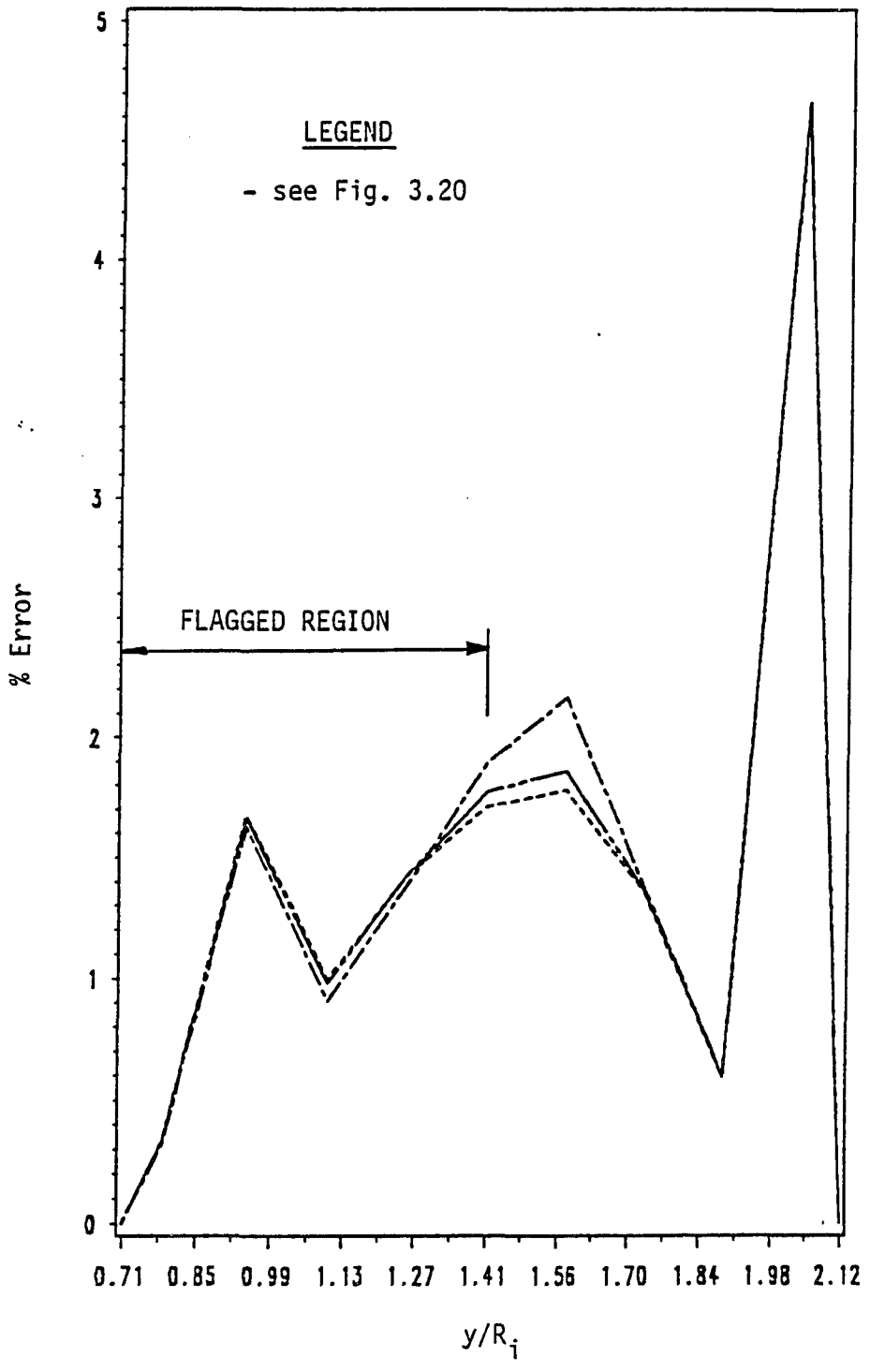


Fig. 3.21 Percent error at x/R_i = 1.0999 for radial conduction in a rotating hollow cylinder. Comparing MAD1-WFDS, MAD2-WDS and MAD3-FDS.

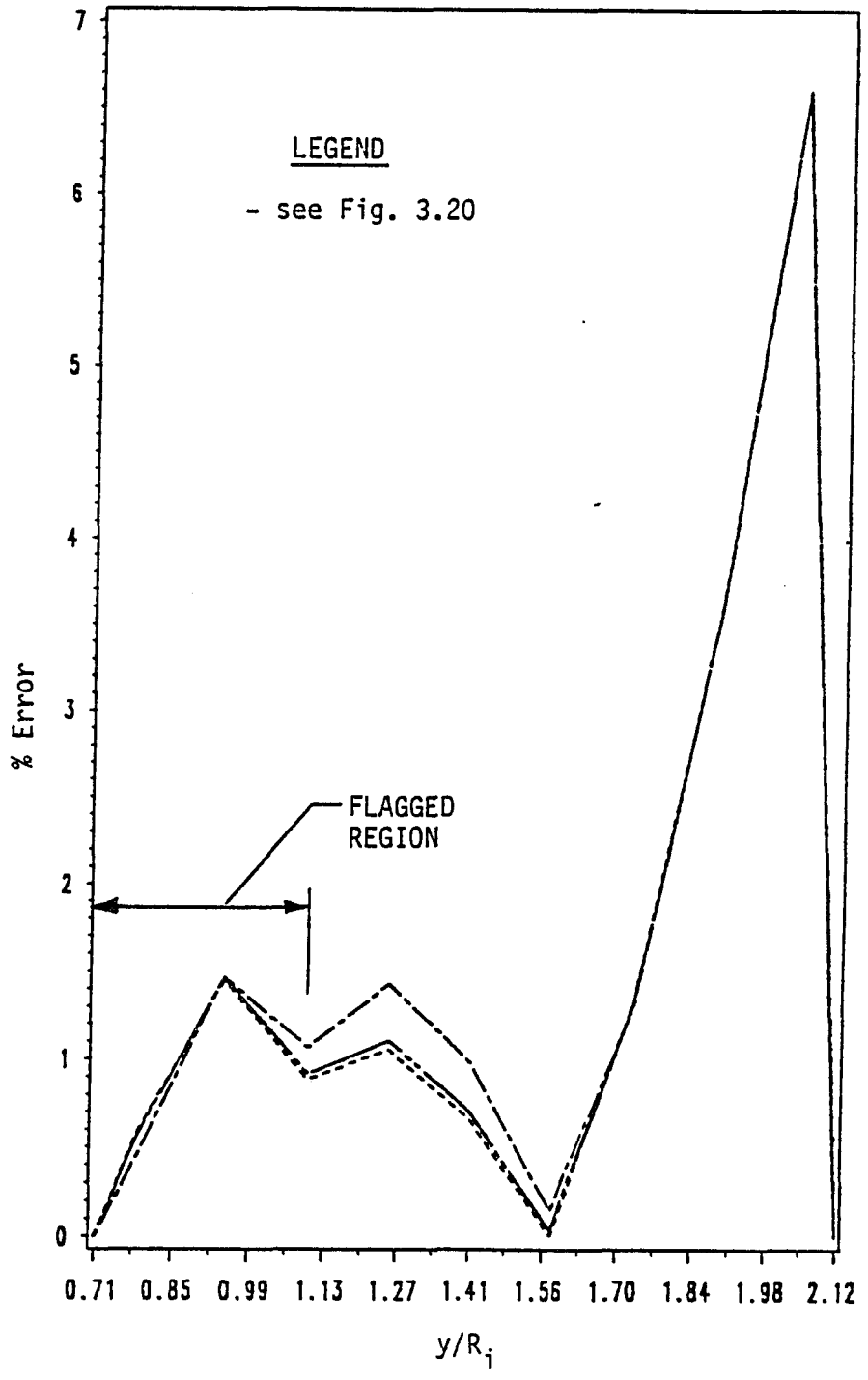


Fig. 3.22 Percent error at x/R_i = 1.4142 for radial conduction in a rotating hollow cylinder. Comparing MAD1-WFDS, MAD2-WDS and MAD3-FDS.

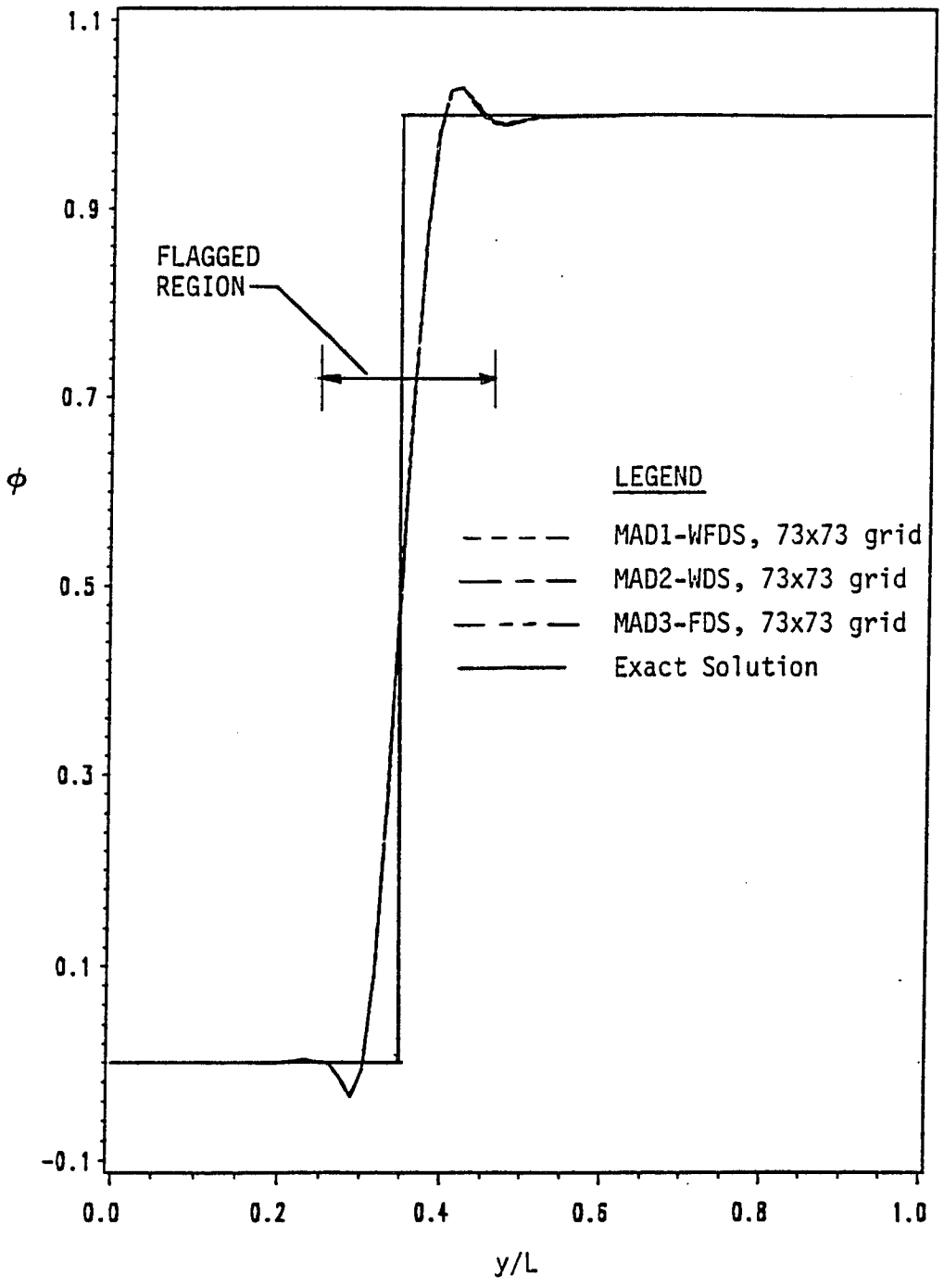


Fig. 3.23 ϕ profile at x/L = 0.2778 for the transport of a step change of a scalar variable in a region with a uniform velocity field. Comparing MAD1-WFDS, MAD2-WDS, MAD3-FDS and exact solutions.

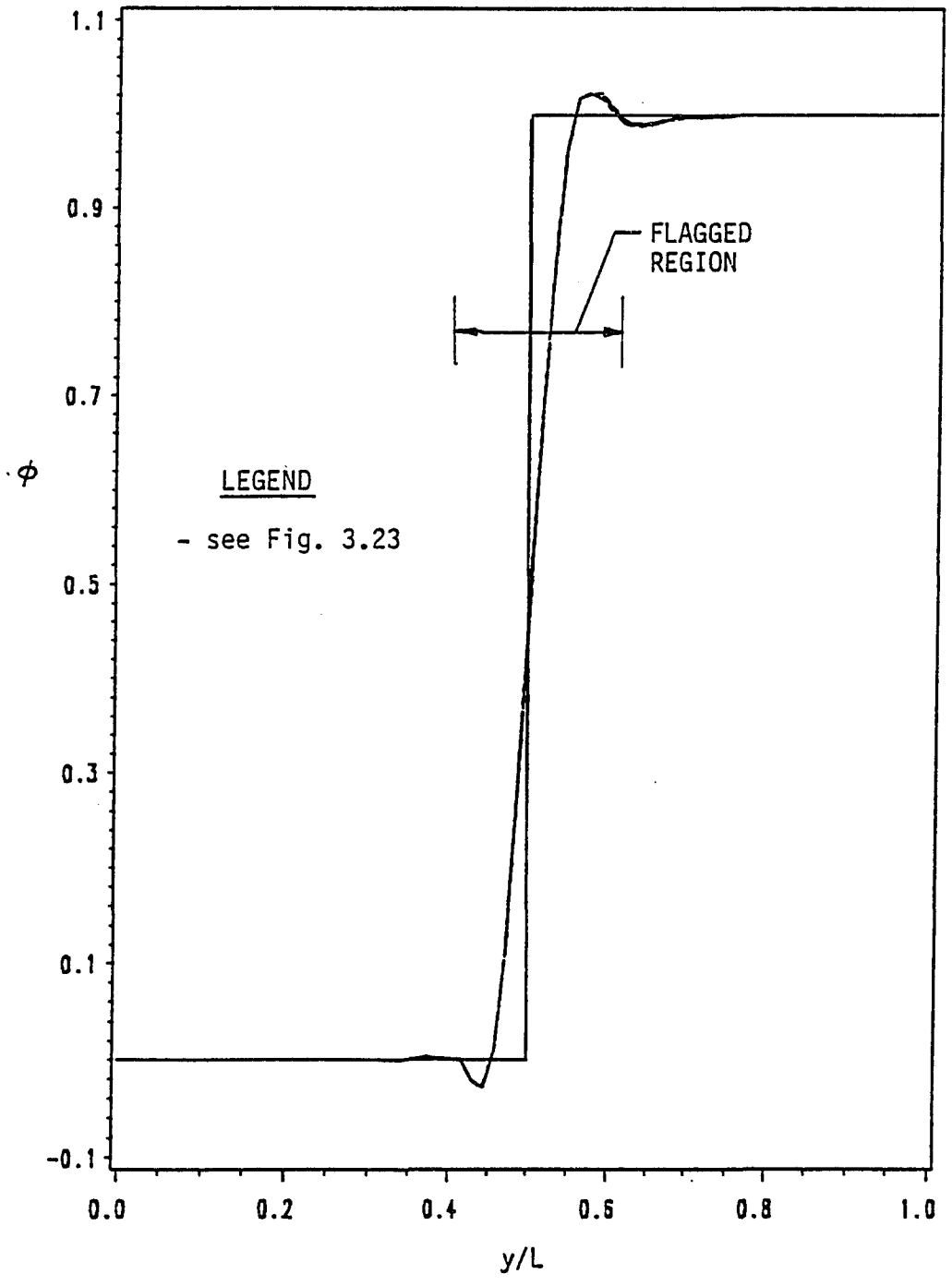


Fig. 3.24 ϕ profile at x/L = 0.5000 for the transport of a step change of a scalar variable in a region with a uniform velocity field. Comparing MAD1-WFDS, MAD2-WDS, MAD3-FDS and exact solutions.

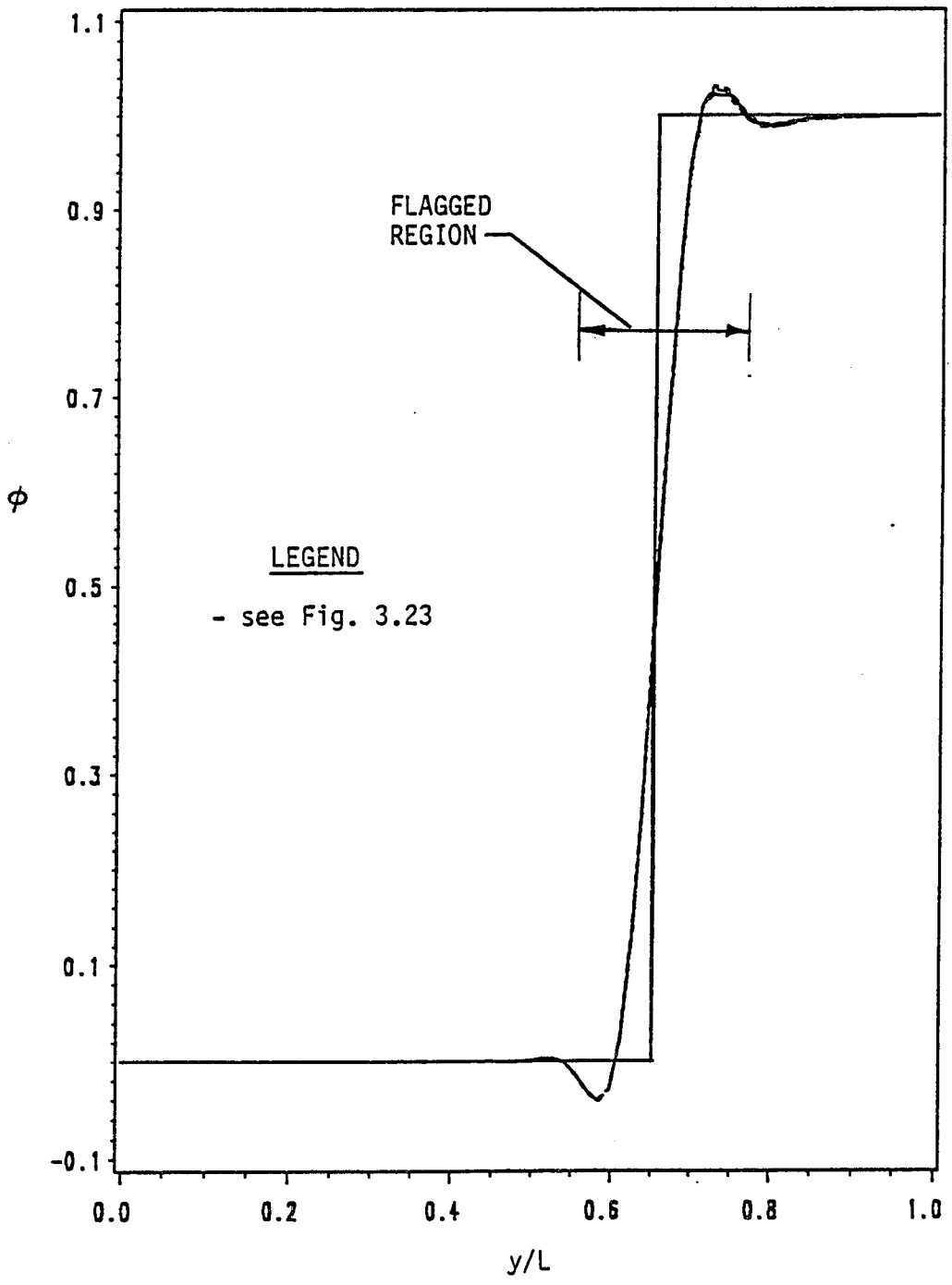


Fig. 3.25 ϕ profile at x/L = 0.7222 for the transport of a step change of a scalar variable in a region with a uniform velocity field. Comparing MAD1-WFDS, MAD2-WDS, MAD3-FDS and exact solutions.

GRID SIZE AND SOLUTION SCHEME	VIRTUAL CPU (sec)	TOTAL CPU (sec)	NO. OF CYCLES	NO. OF ITER	cpu/ ITER
11x11 grid, Upwind	0.35	0.38	---	55	0.0069
QUICK	0.24	0.27	---	30	0.0090
MAD1-WFDS	1.06	1.19	3	165	0.0072
MAD2-WDS	0.60	0.62	35	90	0.0069
MAD3-FDS	0.81	0.87	85	140	0.0062
29x29 grid, Upwind	5.75	5.80	---	125	0.0464

Table 3.1 Computer timing information for radial conduction in a rotating hollow cylinder problem.

GRID SIZE AND SOLUTION SCHEME	VIRTUAL CPU (sec)	TOTAL CPU (sec)	NO. OF CYCLE	NO. OF ITER	cpu/ ITER
11X11 grid, Upwind	0.27	0.29	---	50	0.0058
QUICK	0.19	0.20	---	25	0.0080
MAD1-WFDS	1.15	1.26	3	190	0.0066
MAD2-WDS	0.69	0.77	70	120	0.0064
MAD3-FDS	0.96	1.05	125	175	0.0060
29x29 grid, Upwind	6.26	6.30	---	100	0.0630
73x73 grid, Upwind	77.66	77.97	---	220	0.3544
QUICK	92.16	92.52	---	150	0.6168
MAD1-WFDS	78.70	79.00	2	230	0.3425

Table 3.2 Computer timing information for step change of a scalar variable problem.

advantage is more clearly seen in the flow problems of the next chapter.

3.6 CLOSING REMARKS

Three multiple-grid, adaptive differencing schemes were developed and tested on the two standard problems presented in Chapter Two. The results of each method was examined and the three results compared with the exact solutions. The computer effort required by each method was also presented and disucssed.

The next chapter extends the solution methodology to flow problems. The three adaptive differencing schemes are applied to two standard flow test case problems.

CHAPTER FOUR

FORMULATION OF AND ADAPTATION OF THE FLOW PROBLEM

In Chapters Two and Three the flow field was assumed to be known. In this chapter the methedology for solving an unknown flow field is developed. The three multilevel adaptive differencing algorithms are applied to two test problems.

Recall that in Chpater Two the staggered grid arrangement for the velocity variables u and v was presented. This staggered grid was adopted to avoid the possibility of checkerboard pressure and velocity fields. The staggered grid is shown in Fig. 4.1.

4.1 MOMENTUM EQUATIONS

The momentum equations to be solved are as follows:

$$\iint_{cv} \frac{\partial}{\partial x}(\rho uu) + \frac{\partial}{\partial y}(\rho vu) = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial x}\left(\Gamma \frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\Gamma \frac{\partial u}{\partial y}\right) + S \quad 4.1$$

and

$$\iint_{cv} \frac{\partial}{\partial x}(\rho uv) + \frac{\partial}{\partial y}(\rho vv) = -\frac{\partial p}{\partial y} + \frac{\partial}{\partial x}\left(\Gamma \frac{\partial v}{\partial x}\right) + \frac{\partial}{\partial y}\left(\Gamma \frac{\partial v}{\partial y}\right) + S \quad 4.2$$

These equations may be cast in the form of the general ϕ variable equation 2.1 where ϕ = u,v with the addition of the pressure gradient term. The diffusive and convective fluxes are caclulated in a similar manner to that presented in

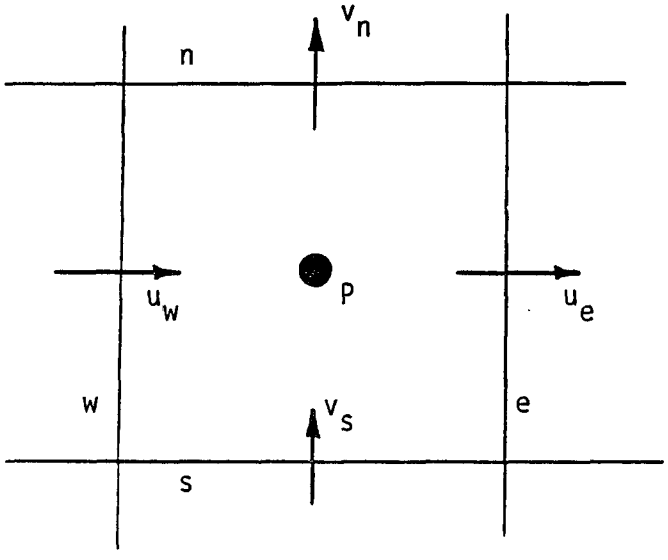


Fig. 4.1a A typical discretized domain for the general variable, ϕ and for pressure, P.

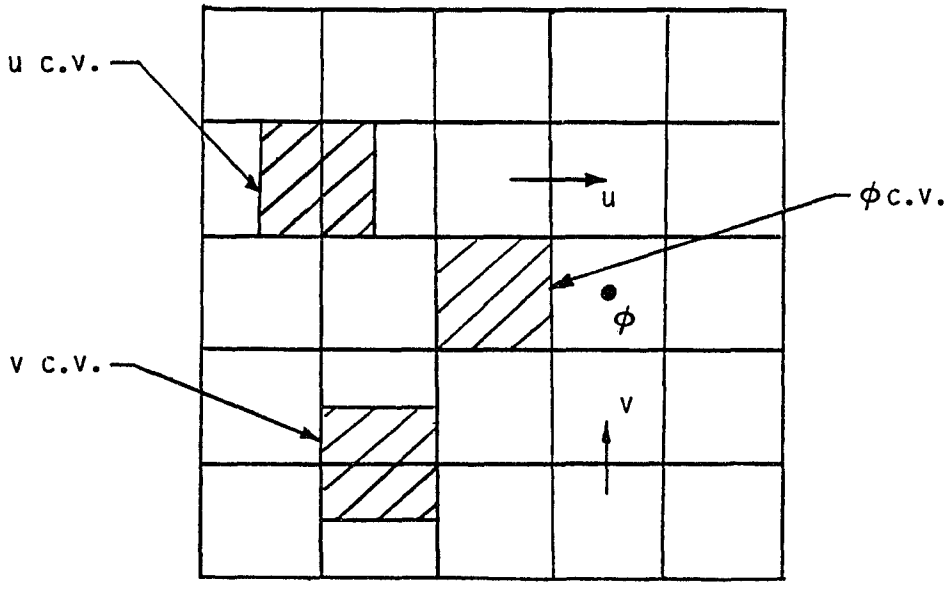


Fig. 4.1b Control volumes for velocity variables.

Chapter Two. The final forms of the resulting discretized governing equations, using the QUICK scheme, are as follows:

$$a_{i,j}\,u_{i,j} = b_{i,j}\,u_{i-1,j} + c_{i,j}\,u_{i+1,j} + d_{i,j}\,u_{i,j+1} + e_{i,j}u_{i,j-1} + f_{i,j} + (p_{i-1,j}-p_{i,j})\,\Delta y_j \qquad 4.3$$

where

$$b_{ij} = F_w\left(\frac{IPOSW}{8}\right)\left[4 + \frac{(\Delta x_{i-1})^2}{\delta x_{i-1}}\left(\frac{1}{\Delta x_{i-1}} + \frac{1}{\Delta x_{i-2}}\right)\right] + F_w\left(\frac{INEGW}{8}\right)\left[4 - \frac{\Delta x_{i-1}}{\delta x_i}\right] + D_w \qquad 4.4a$$

$$c_{i,j} = -0.5\,F_e + D_e \qquad 4.4b$$

$$d_{ij} = F_s\left(\frac{IPOSS}{8}\right)\left[4 + \frac{(\delta y_j)^2}{\Delta y_{j-1}}\left(\frac{1}{\delta y_j} + \frac{1}{\delta y_{j-1}}\right)\right] + F_s\left(\frac{INEGS}{8}\right)\left[4 - \frac{\delta y_j}{\Delta y_j}\right] + D_s \qquad 4.4c$$

$$e_{ij} = -0.5\,F_n + D_n \qquad 4.4d$$

$$a_{ij} = F_e\left(\frac{IPOSE}{8}\right)\left[4 + \frac{(\Delta x_i)^2}{\delta x_i}\left(\frac{1}{\Delta x_i} + \frac{1}{\Delta x_{i-1}}\right)\right] + F_e\left(\frac{INEGE}{8}\right)\left[4 - \frac{\Delta x_i}{\delta x_{i+1}}\right] - 0.5\,F_w$$

$$+ F_n\left(\frac{IPOSN}{8}\right)\left[4 + \frac{(\delta y_{j+1})^2}{\Delta y_j}\left(\frac{1}{\delta y_j} + \frac{1}{\delta y_{j+1}}\right)\right]$$

$$+ F_n\left(\frac{INEGN}{8}\right)\left[4 - \frac{\delta y_{j+1}}{\Delta y_{j+1}}\right] - 0.5F_s$$

$$+ D_e + D_w + D_n + D_s + S_p(\delta x_i)\,\Delta y_j \qquad 4.4e$$

$$f_{ij} = S_c\,\delta x_i\,\Delta y_j + F_e\left(\frac{IPOSE}{8}\right)\frac{(\Delta x_i)^2}{\delta x_i}\left[\frac{u_{i+1,j}}{\Delta x_i} + \frac{u_{i-1,j}}{\Delta x_{i-1}}\right]$$

$$+ F_e\left(\frac{INEGE}{8}\right)\frac{(\Delta x_i)^2}{\delta x_{i+1}}\left[\frac{u_{i+2,j}}{\Delta x_{i+1}} - u_{i+1,j}\left(\frac{1}{\Delta x_{i+1}} + \frac{1}{\Delta x_i}\right)\right]$$

$$- F_w\left(\frac{IPOSW}{8}\right)\frac{(\Delta x_{i-1})^2}{\delta x_{i-1}}\left[\frac{u_{i-2,j}}{\Delta x_{i-2}} + \frac{u_{i,j}}{\Delta x_{i-1}}\right]$$

$$- F_w\left(\frac{INEGW}{8}\right)\frac{(\Delta x_{i-1})^2}{\delta x_i}\left[\frac{u_{i+1,j}}{\Delta x_i} - u_{i,j}\left(\frac{1}{\Delta x_{i-1}} + \frac{1}{\Delta x_i}\right)\right]$$

$$+ F_n\left(\frac{IPOSN}{8}\right)\frac{(\delta y_{j+1})^2}{\Delta y_j}\left[\frac{u_{i,j-1}}{\delta y_j} + \frac{u_{i,j+1}}{\delta y_{j+1}}\right]$$

$$+ F_n\left(\frac{INEGN}{8}\right)\frac{(\delta y_{j+1})^2}{\Delta y_{j+1}}\left[\frac{u_{i,j+2}}{\delta y_{j+2}} - u_{i,j+1}\left(\frac{1}{\delta y_{j+1}} + \frac{1}{\delta y_{j+2}}\right)\right]$$

$$- F_s\left(\frac{IPOSS}{8}\right)\frac{(\delta y_j)^2}{\Delta y_{j-1}}\left[\frac{u_{i,j-2}}{\delta y_{j-1}} + \frac{u_{i,j}}{\delta y_j}\right]$$

$$- F_s\left(\frac{INEGS}{8}\right)\frac{(\delta y_j)^2}{\Delta y_j}\left[\frac{u_{i,j+1}}{\delta y_{j+1}} - u_{i,j}\left(\frac{1}{\delta y_{j+1}} + \frac{1}{\delta y_j}\right)\right]$$

$$- F_e\left(\frac{IPOSE}{24}\right)(\Delta y_j)\left[\frac{u_{i,j+1}}{\delta y_{j+1}} + \frac{u_{i,j-1}}{\delta y_j} - u_{i,j}\left(\frac{1}{\delta y_{j+1}} + \frac{1}{\delta y_j}\right)\right]$$

$$- F_e\left(\frac{INEGE}{24}\right)(\Delta y_j)\left[\frac{u_{i+1,j+1}}{\delta y_{j+1}} + \frac{u_{i+1,j-1}}{\delta y_j} - u_{i+1,j}\left(\frac{1}{\delta y_{j+1}} + \frac{1}{\delta y_j}\right)\right]$$

$$+ F_w\left(\frac{IPOSW}{24}\right)(\Delta y_j)\left[\frac{u_{i-1,j+1}}{\delta y_{j+1}} + \frac{u_{i-1,j-1}}{\delta y_j} - u_{i-1,j}\left(\frac{1}{\delta y_{j+1}} + \frac{1}{\delta y_j}\right)\right]$$

$$+ F_w\left(\frac{INEGW}{24}\right)(\Delta y_j)\left[\frac{u_{i,j+1}}{\delta y_{j+1}} + \frac{u_{i,j-1}}{\delta y_j} - u_{i,j}\left(\frac{1}{\delta y_{j+1}} + \frac{1}{\delta y_j}\right)\right]$$

$$- F_n\left(\frac{IPOSN}{24}\right)(\delta x_i)\left[\frac{u_{i+1,j}}{\Delta x_i} + \frac{u_{i-1,j}}{\Delta x_{i-1}} - u_{i,j}\left(\frac{1}{\Delta x_i} + \frac{1}{\Delta x_{i-1}}\right)\right]$$

$$- F_n\left(\frac{INEGN}{24}\right)(\delta x_i)\left[\frac{u_{i+1,j+1}}{\Delta x_i} + \frac{u_{i-1,j+1}}{\Delta x_{i-1}} - u_{i,j+1}\left(\frac{1}{\Delta x_i} + \frac{1}{\Delta x_{i-1}}\right)\right]$$

$$+ F_s\left(\frac{IPOSS}{24}\right)(\delta x_i)\left[\frac{u_{i+1,j-1}}{\Delta x_i} + \frac{u_{i-1,j-1}}{\Delta x_{i-1}} - u_{i,j-1}\left(\frac{1}{\Delta x_i} + \frac{1}{\Delta x_{i-1}}\right)\right]$$

$$+ F_s\left(\frac{INEGS}{24}\right)(\delta x_i)\left[\frac{u_{i+1,j}}{\Delta x_i} + \frac{u_{i-1,j}}{\Delta x_{i-1}} - u_{i,j}\left(\frac{1}{\Delta x_i} + \frac{1}{\Delta x_{i-1}}\right)\right] \qquad 4.4f$$

As in Chapter Two,

$$F \equiv \rho u(\delta y_j) \qquad or \qquad F \equiv \rho v(\Delta x_i)$$

$$D \equiv \mu\frac{\delta y}{\delta x} \qquad or \qquad D \equiv \mu\frac{\Delta x}{\Delta y}$$

The discretized form of the y-momentum equation is as follows:

$$a_{i,j}v_{i,j} = b_{i,j}v_{i-1,j} + c_{i,j}v_{i+1,j} + d_{i,j}v_{i,j+1} + e_{i,j}v_{i,j-1}$$

$$+ f_{i,j} + (p_{i,j-1} - p_{i,j})\Delta x_i \qquad 4.5$$

where

$$b_{ij} = F_w\left(\frac{IPOSW}{8}\right)\left[4 + \frac{(\delta x_i)^2}{\Delta x_{i-1}}\left(\frac{1}{\delta x_i} + \frac{1}{\delta x_{i-1}}\right)\right] + F_w\left(\frac{INEGW}{8}\right)\left[4 - \frac{\delta x_i}{\Delta x_i}\right] + D_w \qquad 4.6a$$

$$c_{ij} = -0.5F_e + D_e \qquad 4.6b$$

$$d_{ij} = F_s\left(\frac{IPOSS}{8}\right)\left[4 + \frac{(\Delta y_{j-1})^2}{\delta y_{j-1}}\left(\frac{1}{\Delta y_{j-1}} + \frac{1}{\Delta y_{j-2}}\right)\right] + F_s\left(\frac{INEGS}{8}\right)\left[4 - \frac{\Delta y_{j-1}}{\delta y_j}\right] + D_s \qquad 4.6c$$

$$e_{ij} = -0.5F_n + D_n \qquad 4.6d$$

$$a_{ij} = F_e\left(\frac{IPOSE}{8}\right)\left[4 + \frac{(\delta x_{i+1})^2}{\Delta x_i}\left(\frac{1}{\delta x_i} + \frac{1}{\delta x_{i+1}}\right)\right] + F_e\left(\frac{INEGE}{8}\right)\left[4 - \frac{\delta x_{i+1}}{\Delta x_{i+1}}\right] - 0.5F_w$$

$$+ F_n\left(\frac{IPOSN}{8}\right)\left[4 + \frac{(\Delta y_j)^2}{\delta y_j}\left(\frac{1}{\Delta y_j} + \frac{1}{\Delta y_{j-1}}\right)\right] + F_n\left(\frac{INEGN}{8}\right)\left[4 - \frac{\Delta y_j}{\delta y_{j+1}}\right] - 0.5F_s + D_e + D_w + D_n + D_s + S_p(\delta x_i)\,\Delta y_j \qquad 4.6e$$

$$f_{ij} = S_c \Delta x_i \delta y_j + F_e\left(\frac{IPOSE}{8}\right) \frac{(\delta x_{i+1})^2}{\Delta x_i}\left[\frac{v_{i+1,j}}{\delta x_{i+1}} + \frac{v_{i-1,j}}{\delta x_i}\right]$$

$$+ F_e\left(\frac{INEGE}{8}\right) \frac{(\delta x_{i+1})^2}{\Delta x_{i+1}}\left[\frac{v_{i+2,j}}{\delta x_{i+2}} - v_{i+1,j}\left(\frac{1}{\delta x_{i+2}} + \frac{1}{\delta x_{i+1}}\right)\right]$$

$$- F_w\left(\frac{IPOSW}{8}\right) \frac{(\delta x_i)^2}{\Delta x_{i-1}}\left[\frac{v_{i-2,j}}{\delta x_{i-1}} + \frac{v_{i,j}}{\delta x_i}\right]$$

$$- F_w\left(\frac{INEGW}{8}\right) \frac{(\delta x_i)^2}{\Delta x_i}\left[\frac{v_{i+1,j}}{\delta x_{i+1}} - v_{i,j}\left(\frac{1}{\delta x_{i+1}} + \frac{1}{\delta x_i}\right)\right]$$

$$+ F_n\left(\frac{IPOSN}{8}\right) \frac{(\Delta y_j)^2}{\delta y_j}\left[\frac{v_{i,j-1}}{\Delta y_{j-1}} + \frac{v_{i,j+1}}{\Delta y_j}\right]$$

$$+ F_n\left(\frac{INEGN}{8}\right) \frac{(\Delta y_j)^2}{\delta y_{j+1}}\left[\frac{v_{i,j+2}}{\Delta y_{j+1}} - v_{i,j+1}\left(\frac{1}{\Delta y_{j+1}} + \frac{1}{\Delta y_j}\right)\right]$$

$$- F_s\left(\frac{IPOSS}{8}\right) \frac{(\Delta y_{j-1})^2}{\delta y_{j-1}}\left[\frac{v_{i,j-2}}{\Delta y_{j-2}} + \frac{v_{i,j}}{\Delta y_{j-1}}\right]$$

$$+ F_s\left(\frac{INEGS}{8}\right) \frac{(\Delta y_{j-1})^2}{\delta y_j}\left[\frac{v_{i,j+1}}{\Delta y_j} - v_{i,j}\left(\frac{1}{\Delta y_j} + \frac{1}{\Delta y_{j-1}}\right)\right]$$

$$- F_e\left(\frac{IPOSE}{24}\right)(\delta y_j)\left[\frac{v_{i,j+1}}{\Delta y_j} + \frac{v_{i,j-1}}{\Delta y_{j-1}} - v_{i,j}\left(\frac{1}{\Delta y_j} + \frac{1}{\Delta y_{j-1}}\right)\right]$$

$$+ F_e\left(\frac{INEGE}{24}\right)(\delta y_j)\left[\frac{v_{i+1,j+1}}{\Delta y_j} + \frac{v_{i+1,j-1}}{\Delta y_{j-1}} - v_{i+1,j}\left(\frac{1}{\Delta y_j} + \frac{1}{\Delta y_{j-1}}\right)\right]$$

$$- F_w\left(\frac{IPOSW}{24}\right)(\delta y_j)\left[\frac{V_{i-1,j+1}}{\Delta y_j} + \frac{V_{i-1,j-1}}{\Delta y_{j-1}} - v_{i-1,j}\left(\frac{1}{\Delta y_j} + \frac{1}{\Delta y_{j-1}}\right)\right]$$

$$+ F_w\left(\frac{INEGW}{24}\right)(\delta y_j)\left[\frac{V_{i,j+1}}{\Delta y_j} + \frac{V_{i,j-1}}{\Delta y_{j-1}} - v_{i,j}\left(\frac{1}{\Delta y_j} + \frac{1}{\Delta y_{j-1}}\right)\right]$$

$$- F_n\left(\frac{IPOSN}{24}\right)(\Delta x_i)\left[\frac{V_{i+1,j}}{\delta x_{i+1}} + \frac{V_{i-1,j}}{\delta x_i} - v_{i,j}\left(\frac{1}{\delta x_{i+1}} + \frac{1}{\delta x_i}\right)\right]$$

$$+ F_n\left(\frac{INEGN}{24}\right)(\Delta x_i)\left[\frac{V_{i+1,j+1}}{\delta x_{i+1}} + \frac{V_{i-1,j+1}}{\delta x_i} - v_{i,j+1}\left(\frac{1}{\delta x_{i+1}} + \frac{1}{\delta x_i}\right)\right]$$

$$+ F_s\left(\frac{IPOSS}{24}\right)(\Delta x_i)\left[\frac{V_{i+1,j-1}}{\delta x_{i+1}} + \frac{V_{i-1,j-1}}{\delta x_i} - v_{i,j-1}\left(\frac{1}{\delta x_{i+1}} + \frac{1}{\delta x_i}\right)\right]$$

$$+ F_s\left(\frac{INEGS}{24}\right)(\Delta x_i)\left[\frac{V_{i+1,j}}{\delta x_{i+1}} + \frac{V_{i-1,j}}{\delta x_i} - v_{i,j}\left(\frac{1}{\delta x_{i+1}} + \frac{1}{\delta x_i}\right)\right] \qquad 4.6f$$

4.2 PRESSURE CORRECTION EQUATION

If the correct pressure field were specified, then the momentum equations could be calculated directly and would satisfy continuity. However, the pressure field, like the velocity field, is not generally known a priori. Although no direct equation for pressure is available, the pressure is indirectly specified by continuity. This problem is overcome by guessing a pressure field p^* and then correcting it so that continuity is satisfied. In this manner, using the guessed pressure field p^*, the inexact velocity field, u^* and v^*, which results is obtained by solving the following equations:

$$a_e u_e^* = \sum a_{nb} u_{nb}^* + b + (p_P^* - p_E^*)\,\Delta y \qquad 4.7a$$

$$a_n v_n^* = \sum a_{nb} v_{nb}^* + b + (p_P^* - p_N^*)\,\Delta x \qquad 4.7b$$

The correct pressure p may be defined as

$$p = p^* + p' \qquad 4.8$$

where p' is the pressure correction. The velocity field is improved in corresponding manner, so that

$$u = u^* + u' \qquad 4.9a$$

$$v = v^* + v' \qquad 4.9b$$

Substituting Eq. 4.9 into 4.3 and then subtracting Eq. 4.4 from the result yields the following equations:

$$a_e u_e' = \sum a_{nb} u_{nb}' + (p_P' - p_E')\,\Delta y \simeq (p_P' - p_E')\,\Delta y \qquad 4.10a$$

$$a_n v_n' = \sum a_{nb} v_{nb}' + (p_P' - p_N')\,\Delta x \simeq (p_P' - p_N')\,\Delta x \qquad 4.10b$$

The pressure correction equations are obtained from the continuity equation,

$$\frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} = 0 \qquad 4.11$$

Integrating this equation over the control volume shown in Fig. 4.1a (since the pressure values are stored at the main

grid point locations for the general variable ϕ) yields the following equation:

$$[(\rho u)_e - (\rho u)_w]\,\Delta y + [(\rho v)_n - (\rho v)_s]\,\Delta x = 0 \qquad 4.12$$

Substituting the velocity correction equations 4.10 for the velocity components in Eq. 4.12 produces the following pressure correction equation:

$$a_P p'_P = a_E p'_E + a_W p'_W + a_S p'_S + a_N p'_N + b \qquad 4.14$$

where

$$a_E = \rho_e \frac{(\Delta y)^2}{a_e} \qquad 4.15a$$

$$a_W = \rho_w \frac{(\Delta y)^2}{a_w} \qquad 4.15b$$

$$a_N = \rho_n \frac{(\Delta y)^2}{a_n} \qquad 4.15c$$

$$a_S = \rho_s \frac{(\Delta y)^2}{a_s} \qquad 4.15d$$

$$a_P = a_E + a_W + a_N + a_S \qquad 4.15e$$

$$b = [(\rho u^*)_w - (\rho u^*)_e]\,\Delta y + [(\rho v^*)_s - (\rho v^*)_n]\,\Delta x \qquad 4.15f$$

The source term, b, is referred to as a "mass source" term since it is indicative of the degree of error of the pressure and velocity fields which satisfies continuity. Neuman boundary conditions are assumed for the pressure correction equations.

4.3 PRESSURE EQUATION

The equation for pressure is obtained by rewriting the momentum Eq. 4.3 in the following form:

$$u_e = \frac{\sum a_{nb} u_{nb} + b}{a_e} + (p_P - p_E)\,\Delta y \qquad 4.16a$$

$$v_n = \frac{\sum a_{nb} v_{nb} + b}{a_n} + (p_P - p_N)\,\Delta x \qquad 4.16b$$

If the following definitions are made

$$\hat{u}_e = \frac{\sum a_{nb} u_{nb} + b}{a_e} \qquad 4.16a$$

$$\hat{v}_n = \frac{\sum a_{nb} v_{nb} + b}{a_n} \qquad 4.16b$$

then Eq. 4.16 may be rewritten as

$$u_e = \hat{u}_e + (p_P - p_E)\,\Delta y \qquad 4.18a$$

$$v_n = \hat{v}_n + (p_P - p_N)\,\Delta x \qquad 4.18b$$

Substituting these values into the continuity equation (Eq. 4.12) yields the following pressure equation:

$$a_P p_P = a_E p_E + a_W p_W + a_S p_S + a_N p_N + b \qquad 4.19$$

where

$$a_E = \rho_e \frac{(\Delta y)^2}{a_e} \qquad 4.19a$$

$$a_W = \rho_w \frac{(\Delta y)^2}{a_w} \qquad 4.19b$$

$$a_N = \rho_n \frac{(\Delta y)^2}{a_n} \qquad 4.19c$$

$$a_S = \rho_s \frac{(\Delta y)^2}{a_s} \qquad 4.19d$$

$$a_P = a_E + a_W + a_N + a_S \qquad 4.19e$$

$$b = [(\rho\hat{u})_w - (\rho\hat{u})_e]\,\Delta y + [(\rho\hat{v})_s - (\rho\hat{v})_n]\,\Delta x \qquad 4.19f$$

The pressure and pressure correction equations are similar, by no approximations were made in deriving the pressure equation. Neuman boundary conditions are adopted for the pressure equation, also.

4.4 SOLUTION PROCEDURE

The algorithm used in this research for solving the velocity, pressure and pressure correction equations was developed by Patankar (1980) and is called the SIMPLER algorithm (Semi-Implicit Method for Pressure Linked Equations - Revised). The algorithm proceeds as follows:

1. Start with a guessed velocity field.
2. Calculate the coefficients for the momentum equations and then $\hat{u}$ and $\hat{v}$ from Eq. 4.16.

3. Calculate the coefficients for the pressure equation and solve the pressure equation to obtain the new pressure field.
4. Treating this pressure field as p^*, solve the momentum equations to obtain the new velocity field, u^* and v^*.
5. Calculate the mass source term b (of Eq. 4.15) and solve the p' equation.
6. Use the p' field to correct the velocity field using Eq. 4.10, but do not update the pressure field.
7. Solve the discretization equations for other general ϕ variables if necessary.
8. Return to Step 2 and repeat until convergence.

4.5 ADAPTATION

The SIMPLER algorithm is used to obtain solutions both on the golbal domain and on any flagged subdomains(s). The convection terms of the momentum equations are discretized using either the upwind or QUICK schemes accordingly.

The three multiple-grid adaptive differencing procedures are essentially the same as presented in Chapter Three. The momentum and general ϕ equations are solved with the QUICK discretization scheme for the convection terms in the flagged subdomains. For MAD1-WFDS, recall that

$$L_0\phi_0 = L_0\phi_1 \qquad \text{in } \Omega_{1,i}$$

after the adptation process has started. Similar treatment is accorded the flow field. The MAD1-WFDS algorithm for an unknown flow field is as follows:

1. In Ω_0, obtain the first order solution

$$L_0 u_0 = b_u$$

$$L_0 v_0 = b_v$$

$$L_0 \phi_0 = b_\phi .$$

2. Flag grid points with error estimates E_n greater than a specified tolerance, and define $\Omega_{1,i}$.

3. In $\Omega_{1,i}$, obtain the third order solution

$$L_1 u_1 = b_u$$

$$L_1 v_1 = b_v$$

$$L_1 \phi_1 = b_\phi$$

with boundary conditions based on u_0, v_0, and ϕ_0, respectively.

4. In Ω_0, obtain improved solutions by solving

in $\Omega_0 - \Omega_{1,i}$:

$$L_0 u_0 = b_u$$

$$L_0 v_0 = b_v$$

$$L_0 \phi_0 = b_\phi .$$

in $\Omega_{1,i}$:

$$L_0 u_0 = L_0 u_1$$

$$L_0 v_0 = L_0 v_1$$

$$L_0 \phi_0 = L_0 \phi_1 .$$

5. Return to Step 3 and repeat until the desired accuracy level is satisfied.
6. Proceed to the next level of adaptive differencing by defining $\Omega_{2,i}$, which will generally be nested in $\Omega_{1,i}$, and repeat above steps. Continue to the desired accuracy levels.

Note that the equations to be solved in region $\Omega_{1,i}$ in Step 4 above may be expressed as follows:

$$L_0 u_0 = a_P^0 u_P^0 - \sum a_{nb}^0 u_{nb}^0 = a_P^0 u_P^1 - \sum a_{nb}^0 u_{nb}^1 = L_0 u_1$$

$$L_0 v_0 = a_P^0 v_P^0 - \sum a_{nb}^0 v_{nb}^0 = a_P^0 v_P^1 - \sum a_{nb}^0 v_{nb}^1 = L_0 v_1$$

$$L_0 \phi_0 = a_P^0 \phi_P^0 - \sum a_{nb}^0 \phi_{nb}^0 = a_P^0 \phi_P^1 - \sum a_{nb}^0 \phi_{nb}^1 = L_0 \phi_1$$

where the superscript 0 indicates quantities computed using the upwind scheme and the superscript 1 denotes quantities calculated employing the QUICK scheme.

Initially, the pressure equation was also modified in the $\Omega_{1,i}$ region after adptation so that

$$L_0 p_0 = b_p \qquad \text{in } (\Omega_0 - \Omega_{1,i})$$

and

$$L_0 p_0 = L_0 p_1 \qquad \text{in } \Omega_{1,i}.$$

However no significant improvement in the results or convergence rate was observed. Therefore, this practice was not adopted. The same flagging criteria is applied to the

velocity fields. In the next section, the three MAD schemes are applied to two test case problems.

4.6 TEST PROBLEMS

Two standard flow problems are studied as test cases. The first problem considered is laminar, isothermal flow in a square cavity that is driven by a moving upper plate. The second problem studied is flow over a backward facing step. The first order upwind and the three multigrid schemes are compared to an "exact" solution obtained by using the QUICK algorithm on a finer mesh.

4.6.1 Driven Flow in a Square Cavity

This problem is widely used in computational fluid dynamics as a benchmark case (see for example, Burggraf, 1966; Ghia et al, 1982: and Schreiber and Keller, 1983). Figure 4.2 shows the problem domain. The governing equations are as follows:

$$\frac{\partial}{\partial X}(U^2) + \frac{\partial}{\partial Y}(UV) = -\frac{\partial P}{\partial X} + \frac{1}{Re}\left[\frac{\partial^2 U}{\partial X^2} + \frac{\partial^2 U}{\partial Y^2}\right] \qquad 4.20$$

$$\frac{\partial}{\partial X}(UV) + \frac{\partial}{\partial Y}(V^2) = -\frac{\partial P}{\partial Y} + \frac{1}{Re}\left[\frac{\partial^2 V}{\partial X^2} + \frac{\partial^2 V}{\partial Y^2}\right] \qquad 4.21$$

and

$$\frac{\partial U}{\partial X} + \frac{\partial V}{\partial Y} = 0 \qquad 4.22$$

where

$$X = \frac{x}{L}, \qquad Y = \frac{y}{L},$$

$$U = \frac{u}{u_{lid}}, \qquad V = \frac{v}{u_{lid}},$$

$$P = \frac{p}{\rho u_{lid}^2}, \qquad Re = \frac{\rho L u_{lid}}{\mu},$$

This problem was solved using a value of 100 for the Reynolds number. The boundary conditions are zero velocity along the three stationary walls and

$$U = 1, \quad \text{and} \quad V = 0$$

at the sliding lid.

The problem is solved on a 41 by 41 grid using the upwind method and the three multigrid methods. A finer grid (79 by 79 mesh size) QUICK solution is obtained and used for comparison purposes along with Burggraf's (1966) solution. Fig. 4.3 shows the discretized domain of a 15 x 15 grid with the flagged region. The 41 x 41 mesh flagged region is proportionately identical. The smaller grid size is used in Fig. 4.3 for clarity.

Figures 4.4 and 4.5 show the results of the MAD1-WFDS and upwind solutions with fine grid QUICK solution and the Burggraf solution. Figure 4.4 displays plots of the dimensionless U velocity and V velocity profiles, respectively, as a function of y/L at x/L = 0.5. The multigrid solution shows a definite improvement over the upwind solution

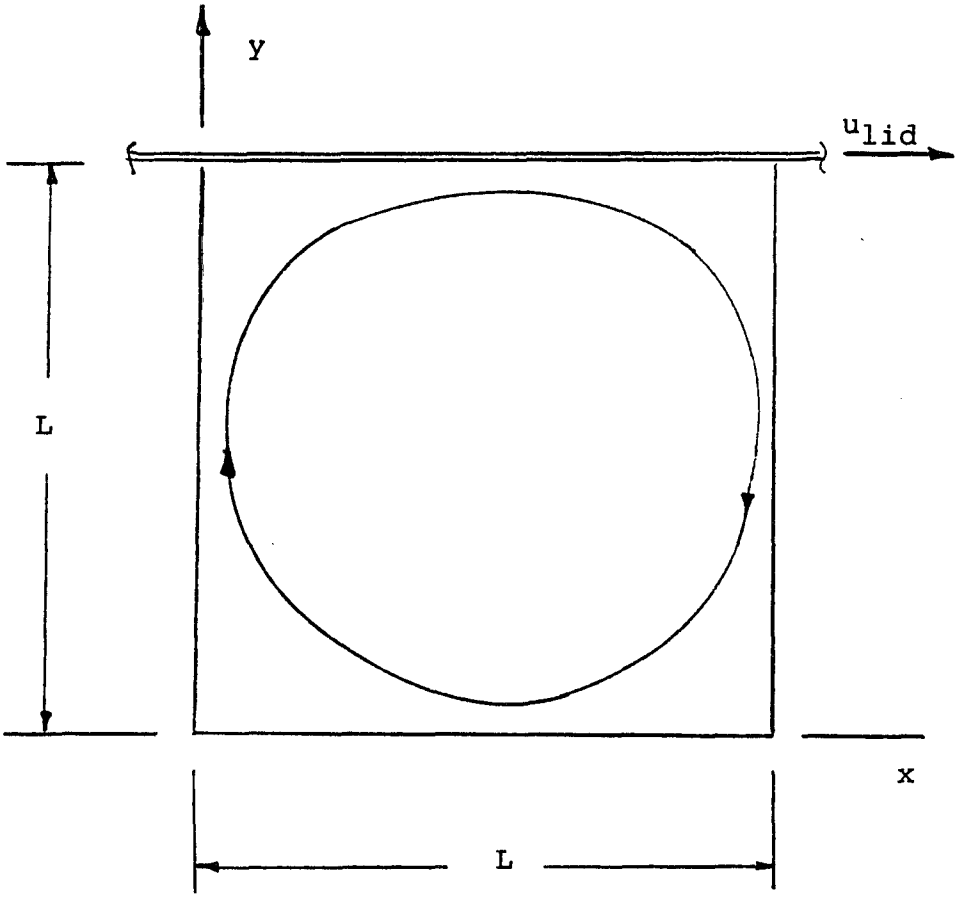


Fig. 4.2 Physical domain driven flow in a square cavity.

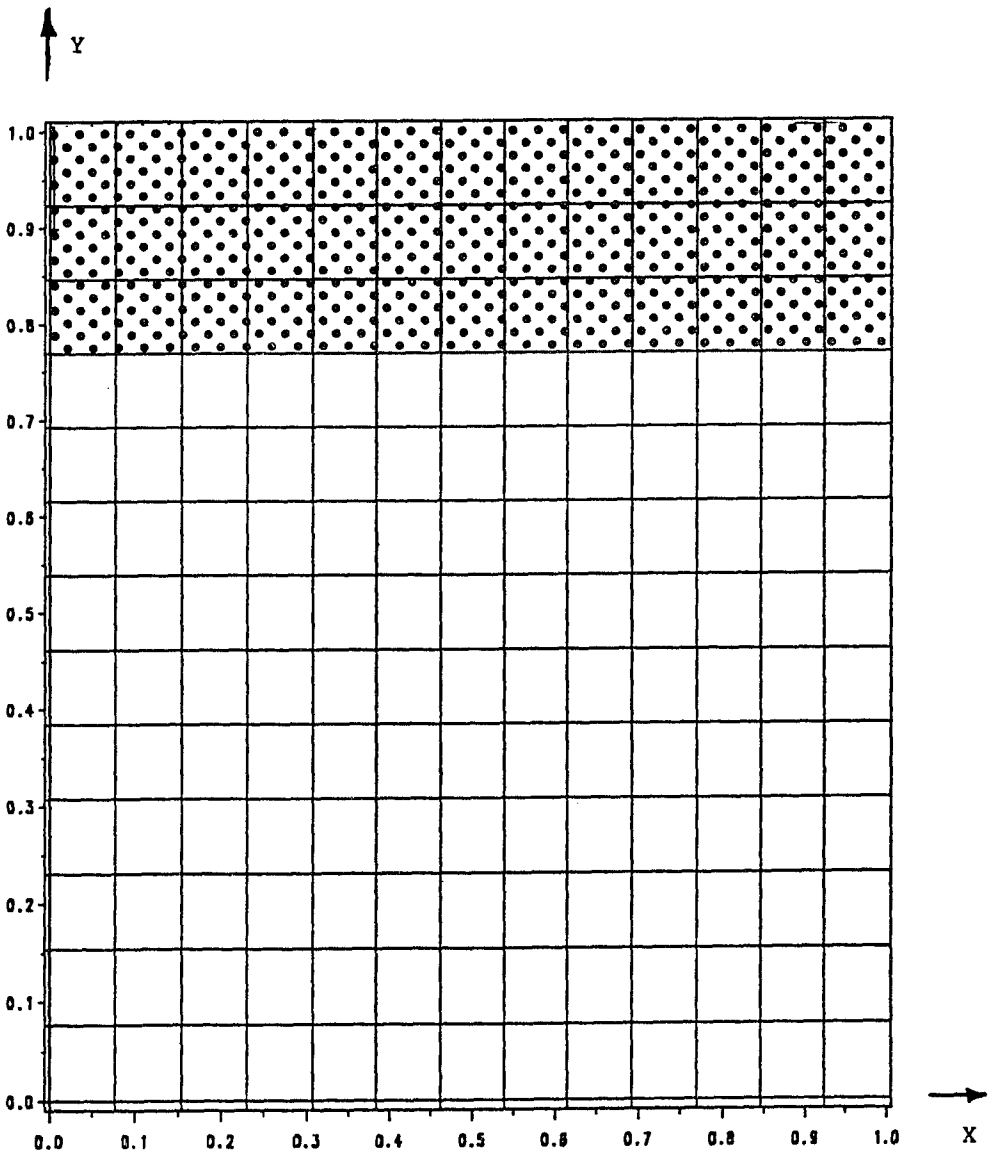


Fig. 4.3 Discretized domain and flagged region for driven flow in a square cavity on a 15 x 15 grid.

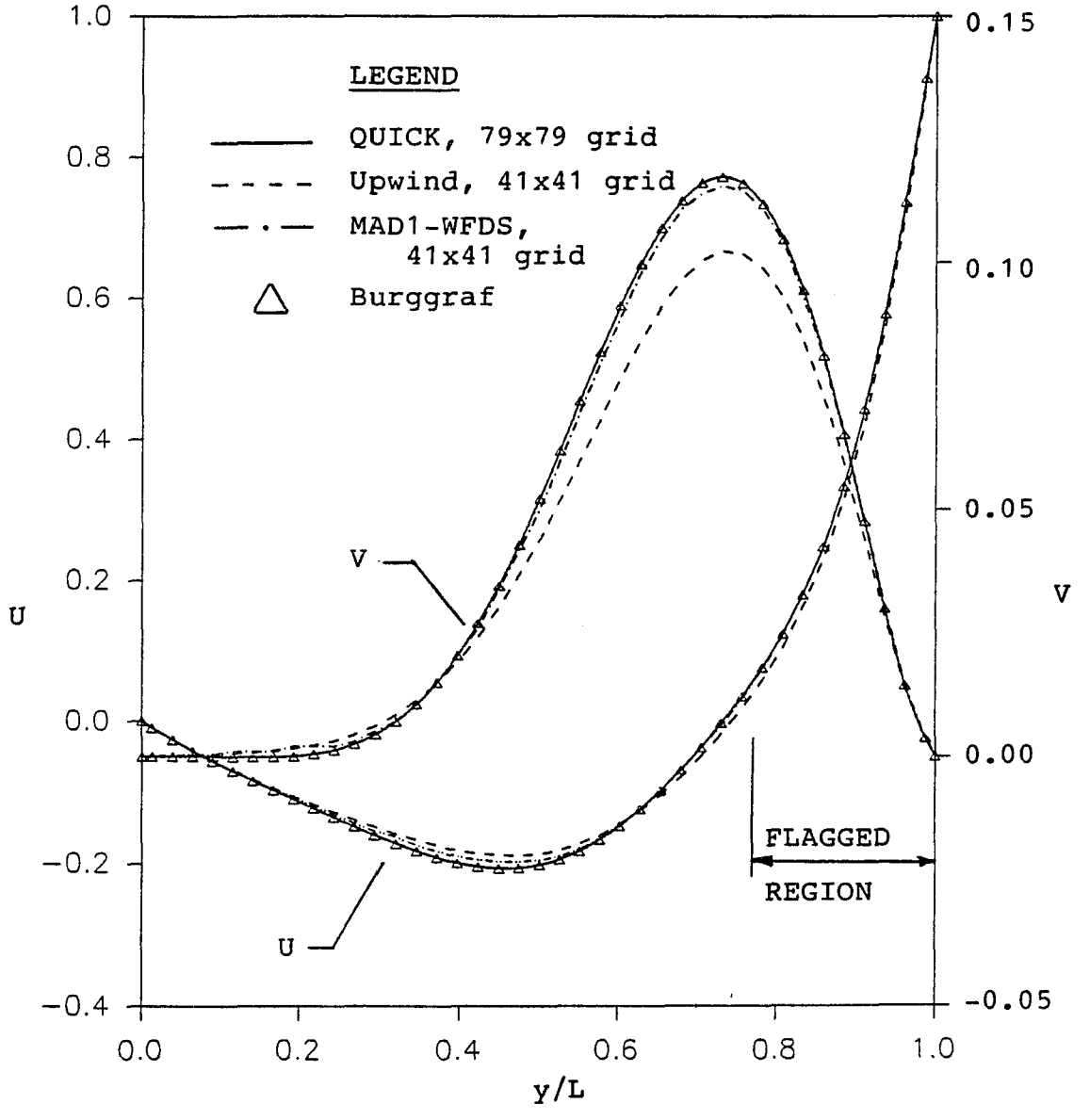


Fig. 4.4 U-velocity and V-velocity profiles at x/L = 0.5 for driven flow in a square cavity. Comparing upwind, MAD1-WFDS, Burggraf and fine grid QUICK solutions.

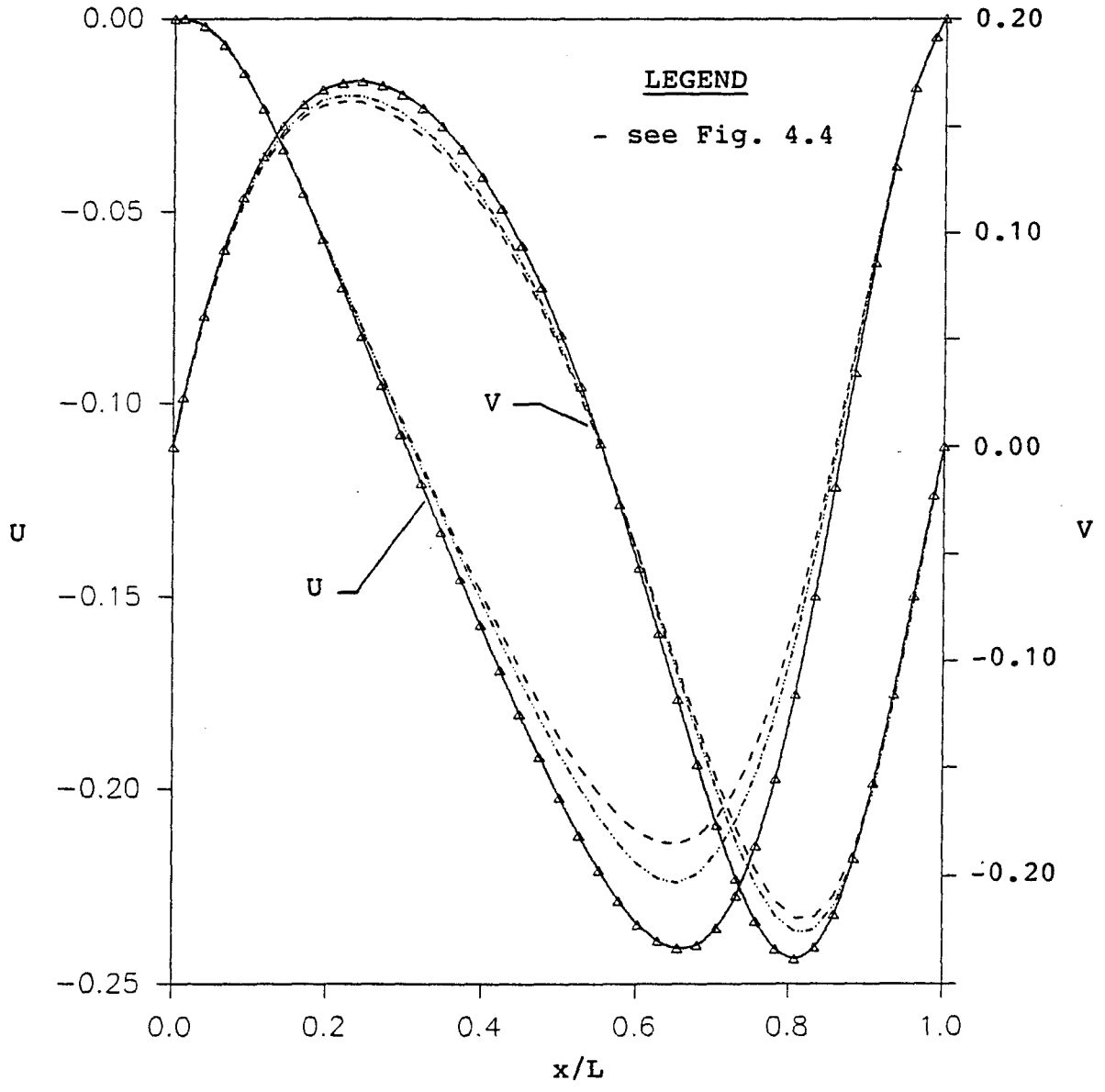


Fig. 4.5 U-velocity and V-velocity profiles at y/L = 0.5 for driven flow in a square cavity. Comparing upwind, MAD1-WFDS, Burggraf and fine grid QUICK solutions.

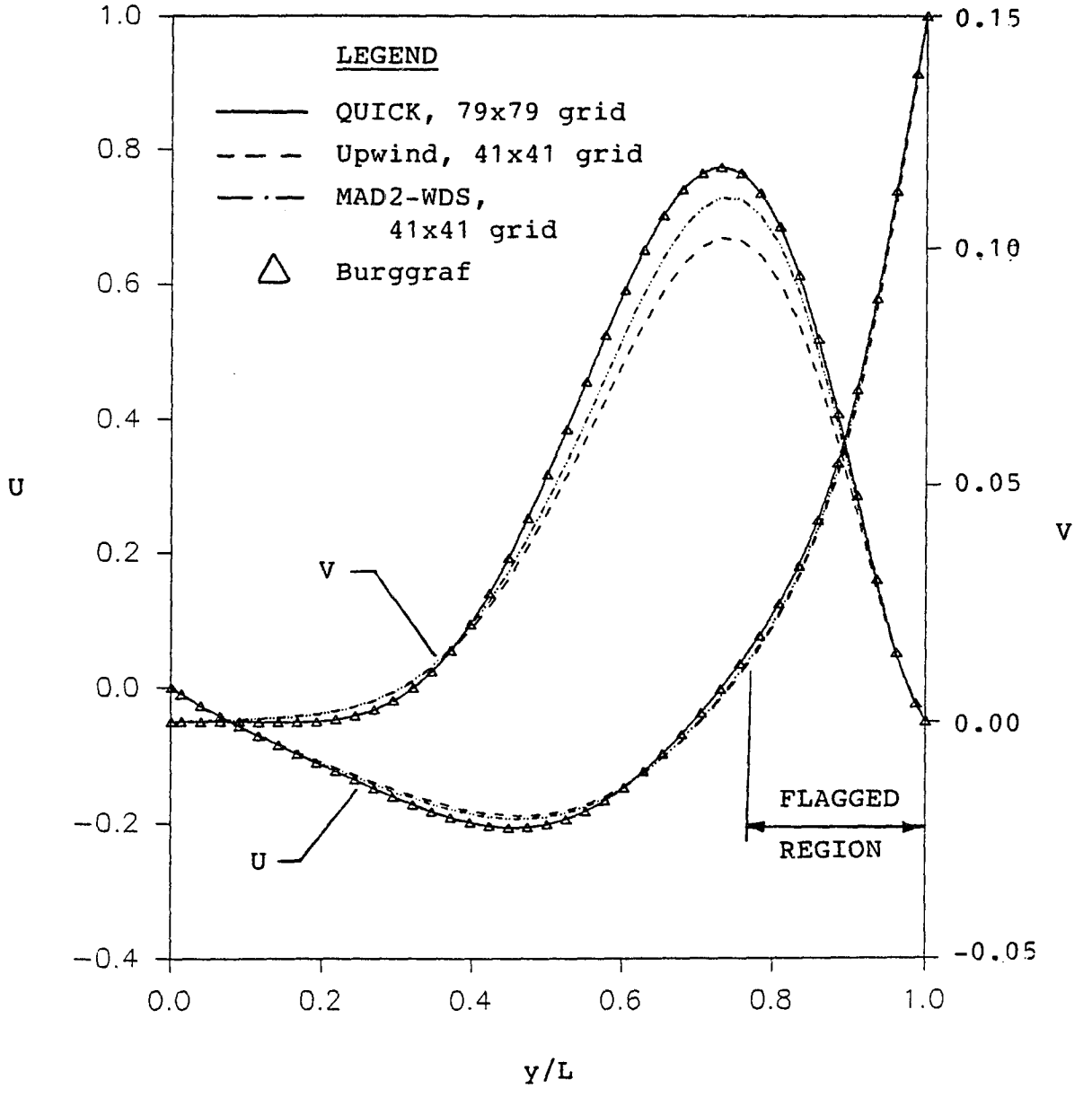


Fig. 4.6 U-velocity and V-velocity profiles at x/L = 0.5 for driven flow in a square cavity. Comparing upwind, MAD2-WDS, Burggraf and fine grid QUICK solutions.

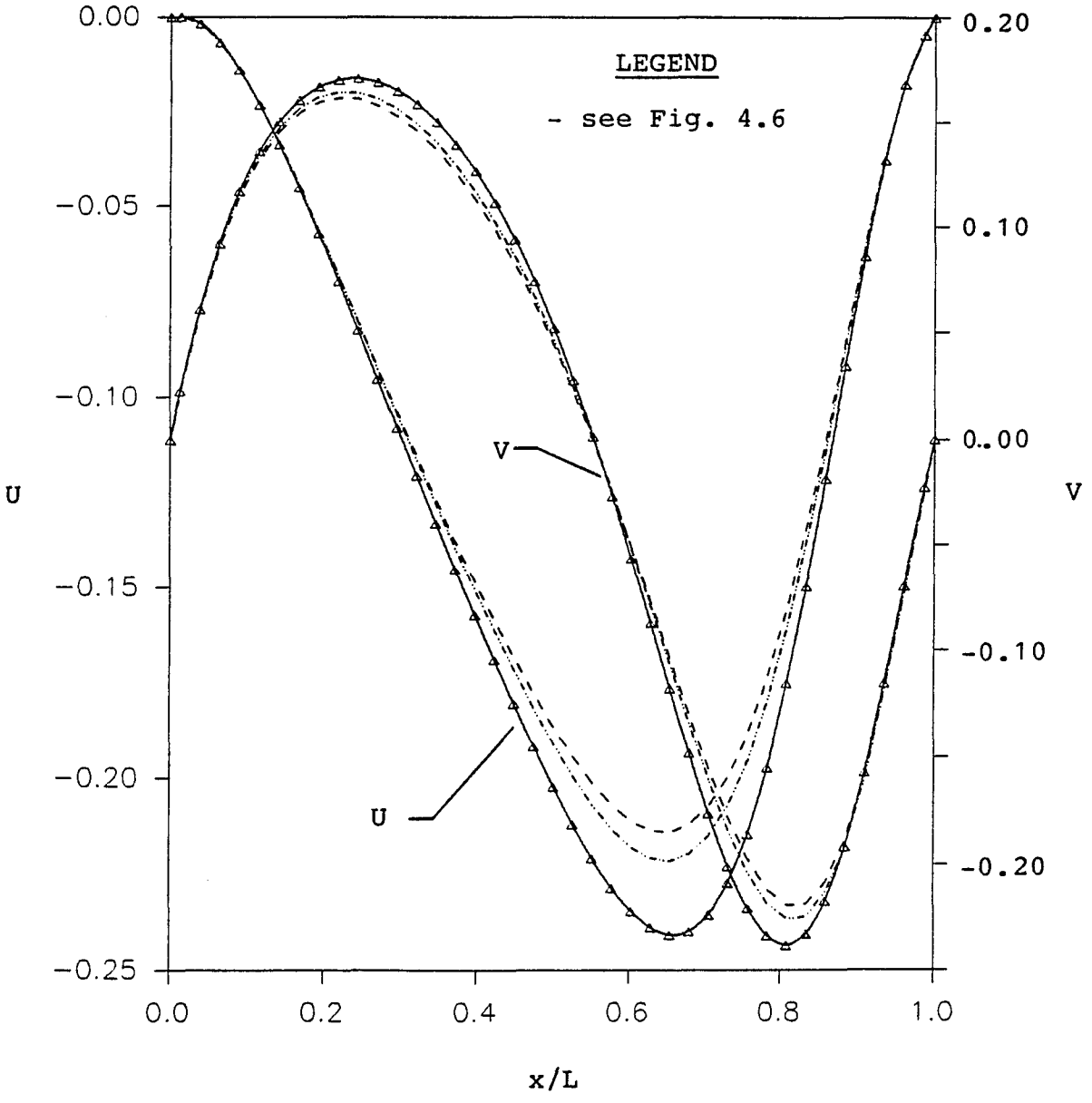


Fig. 4.7 U-velocity and V-velocity profiles at y/L = 0.5 for driven flow in a square cavity. Comparing upwind, MAD2-WDS, Burggraf and fine grid QUICK solutions.

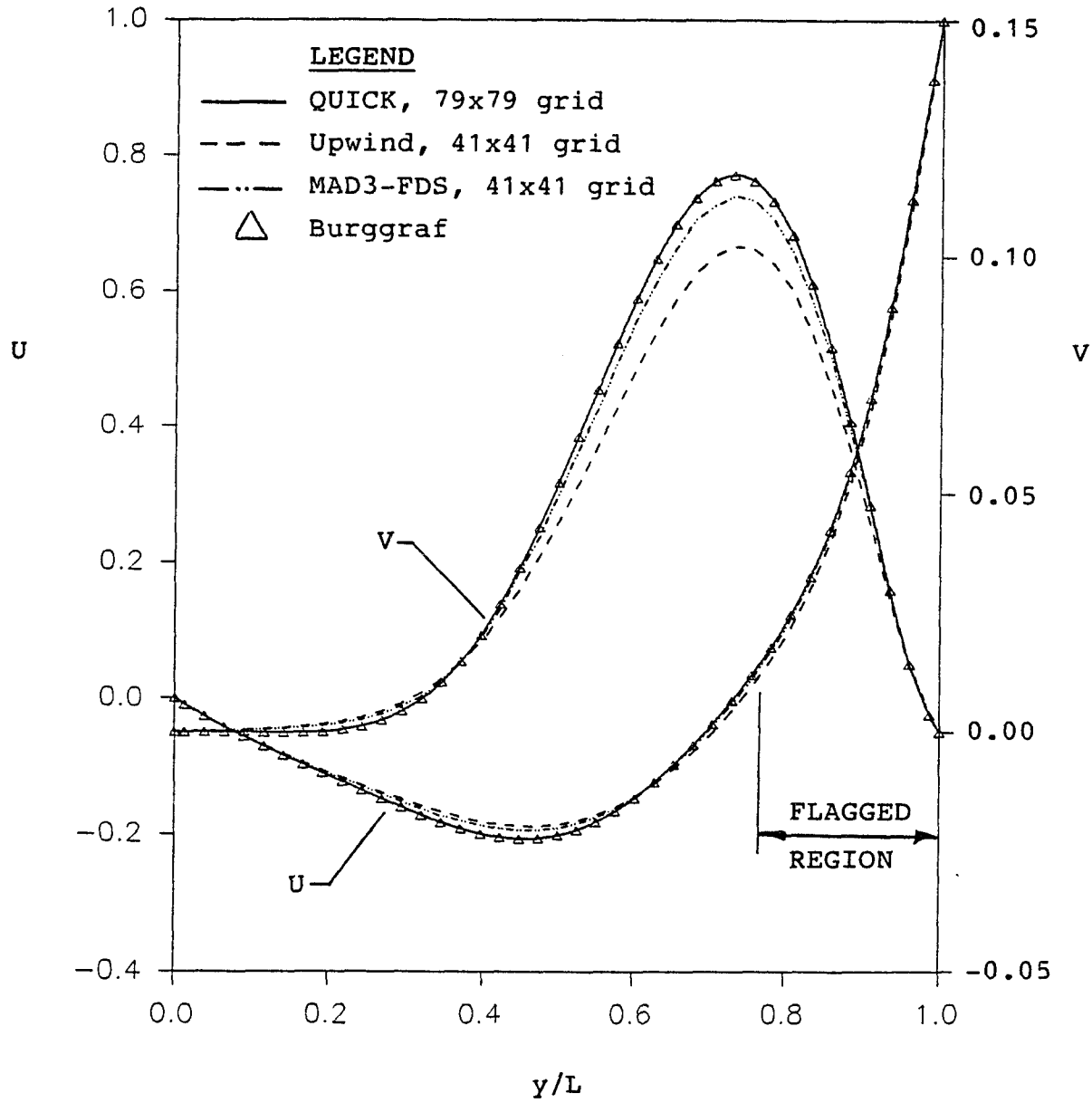


Fig. 4.8 U-velocity and V-velocity profiles at x/L = 0.5 for driven flow in a square cavity. Comparing upwind, MAD3-FDS, Burggraf and fine grid QUICK solutions.

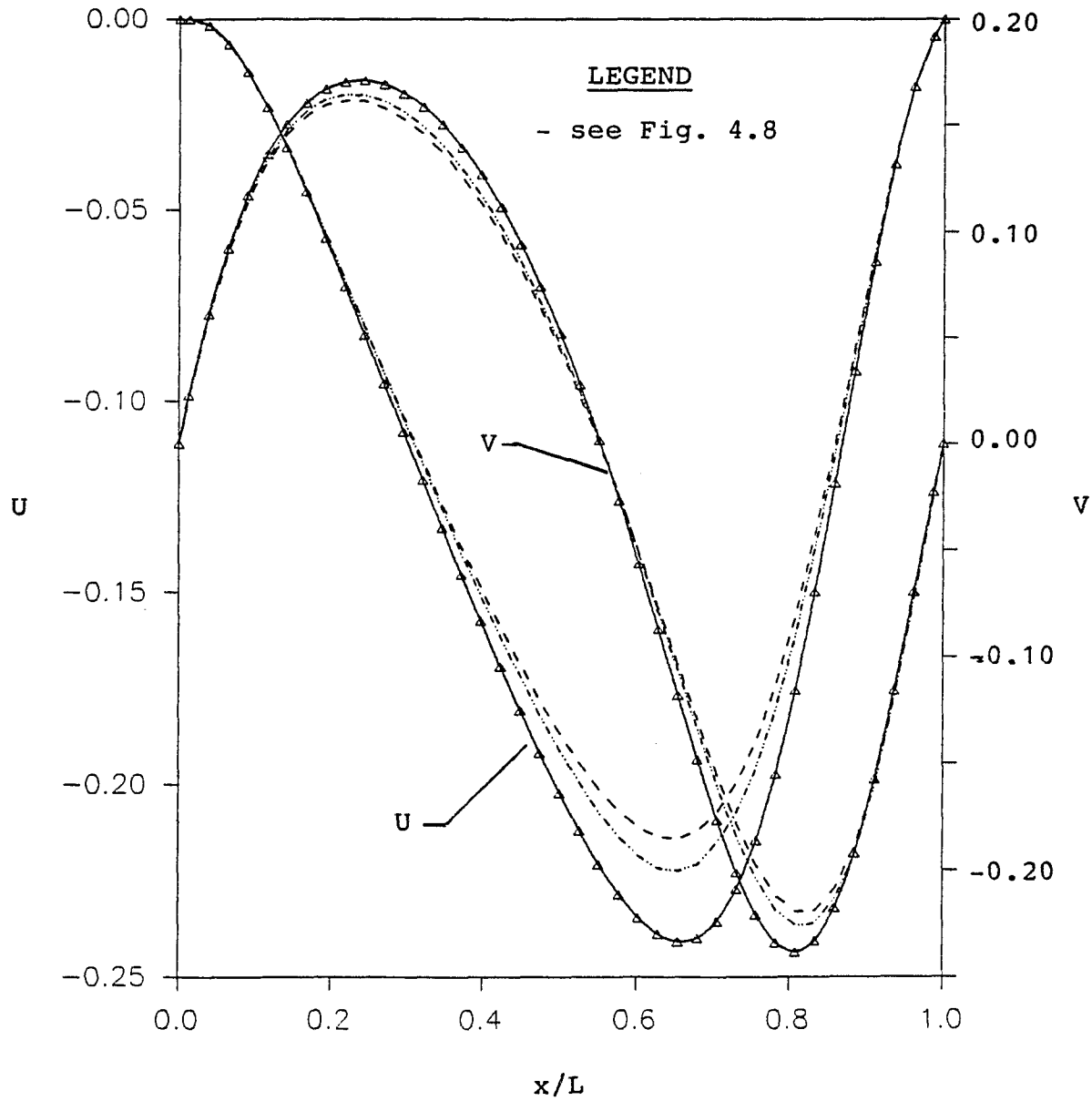


Fig. 4.9 U-velocity and V-velocity profiles at y/L = 0.5 for driven flow in a square cavity. Comparing upwind, MAD3-FDS, Burggraf and fine grid QUICK solutions.

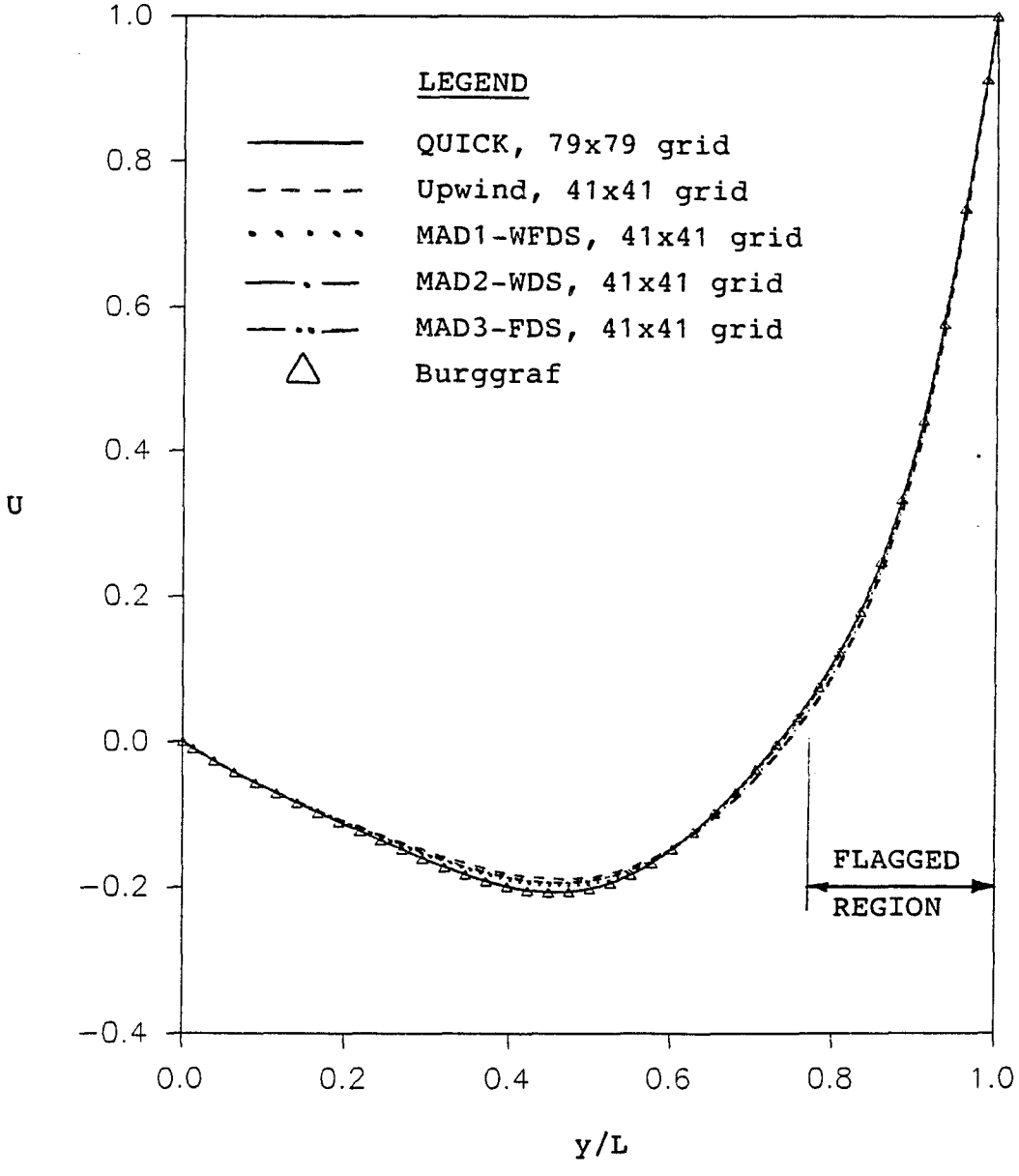


Fig. 4.10 U-velocity profile at x/L = 0.5 for driven flow in a square cavity. Comparing upwind, MAD1-WFDS, MAD2-WDS, MAD3-FDS, Burggraf and fine grid QUICK solutions.

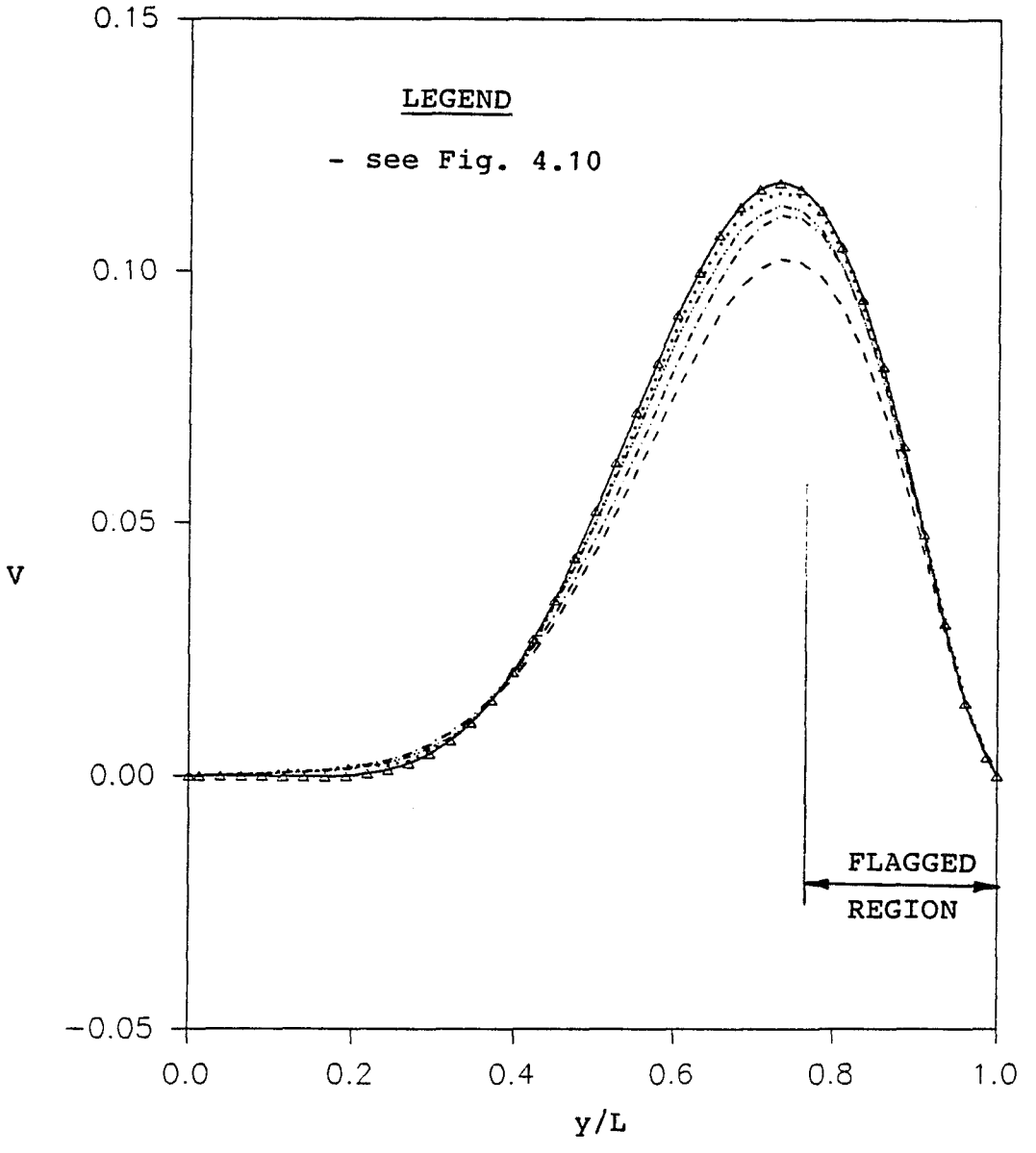


Fig. 4.11 V-velocity profile at x/L = 0.5 for driven flow in a square cavity. Comparing upwind, MAD1-WFDS, MAD2-WDS, MAD3-FDS, Burggraf and fine grid QUICK solutions.

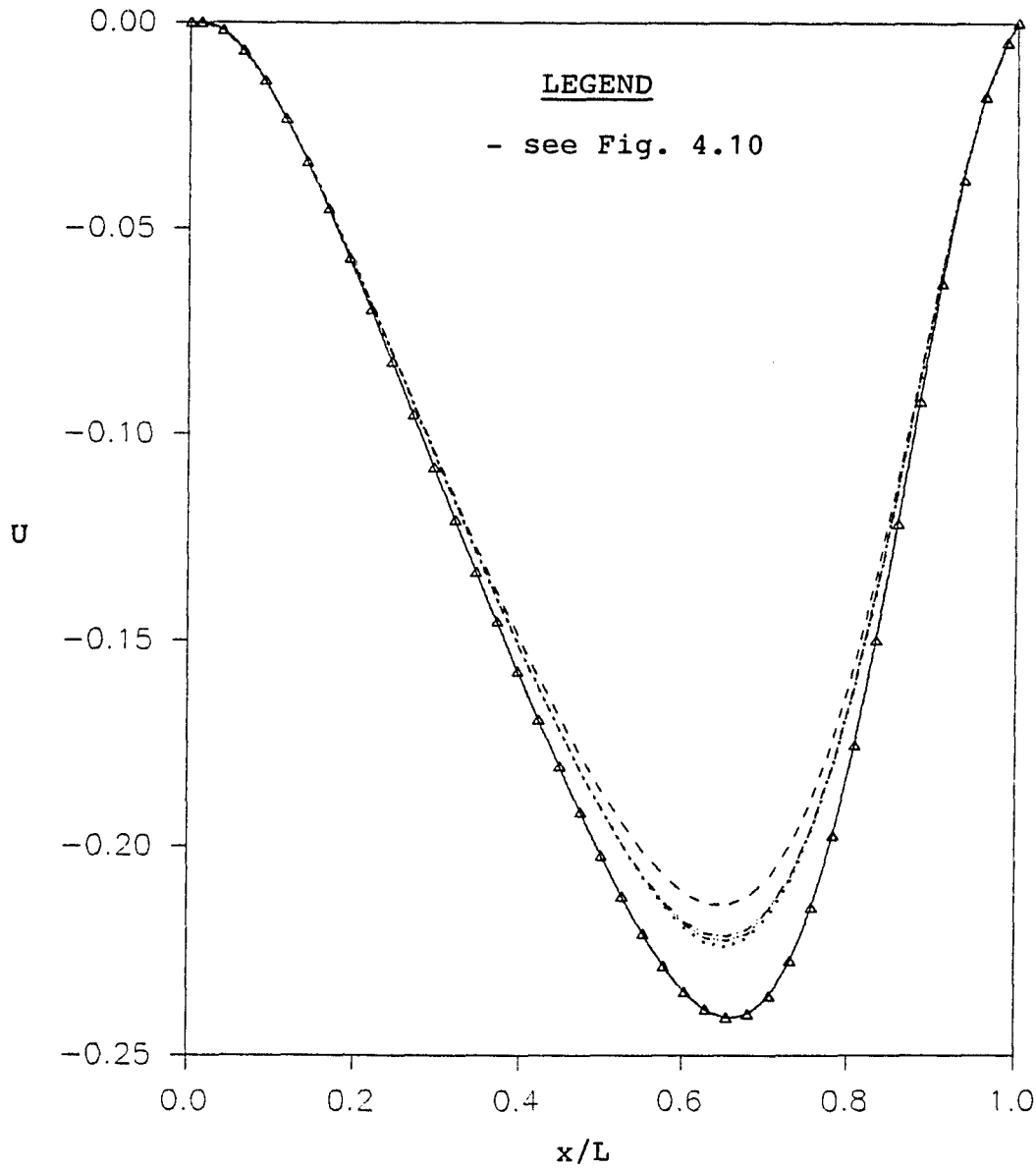


Fig. 4.12 U-velocity profile at y/L = 0.5 for driven flow in a square cavity. Comparing upwind, MAD1-WFDS, MAD2-WDS, MAD3-FDS, Burggraf and fine grid QUICK solutions.

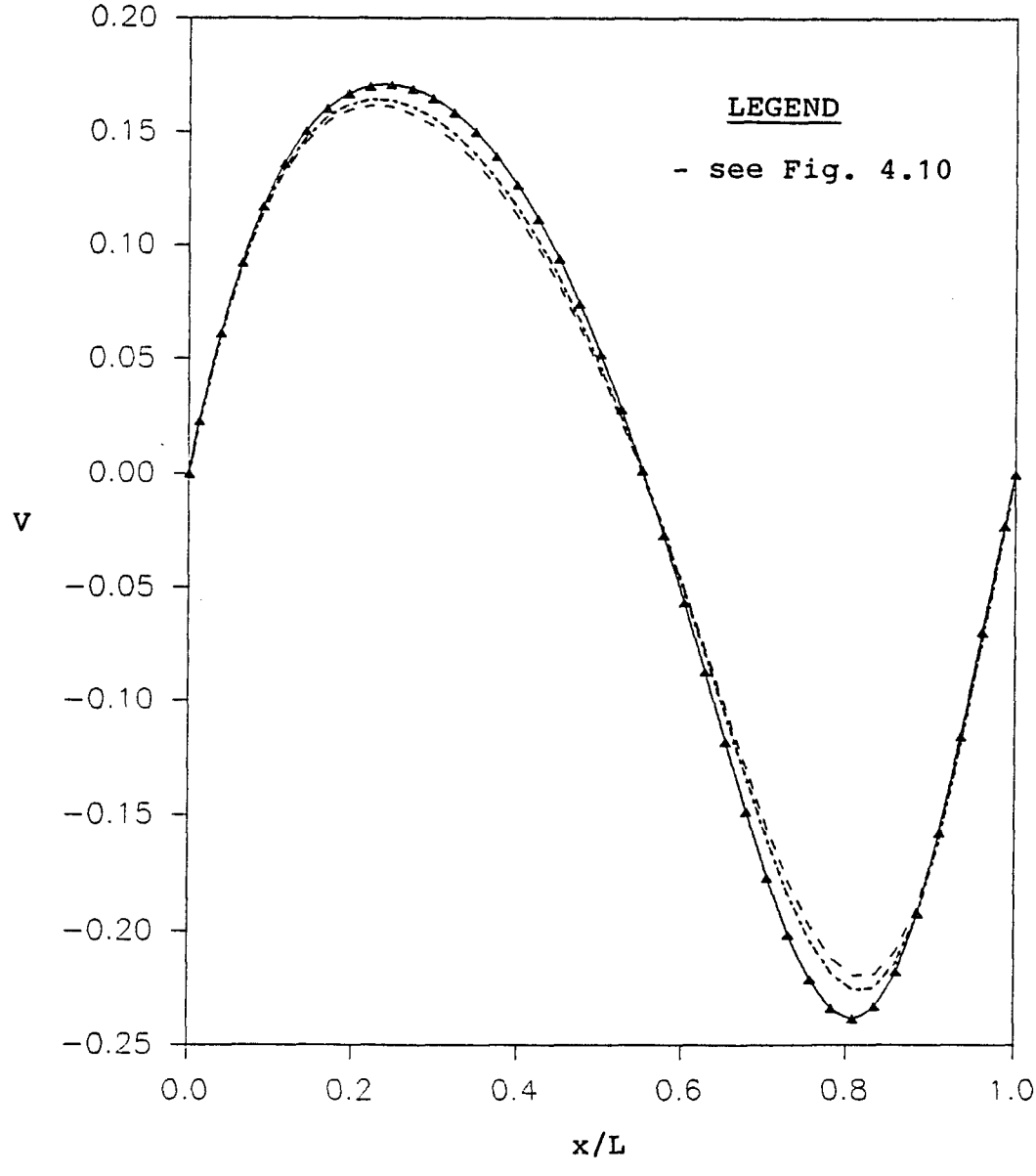


Fig. 4.13 V-velocity profile at y/L = 0.5 for driven flow in a square cavity. Comparing upwind, MAD1-WFDS, MAD2-WDS, MAD3-FDS, Burggraf and fine grid QUICK solutions.

even outside of the flagged region. If percent error is defined as follows,

$$\% \text{ error} = 100 \times (\text{QUICK} - Z) / \text{QUICK}$$

(where Z may be defined as the upwind or any of the MAD schemes) then at y/L = 0.731 the MAD1-WFDS solution for the V velocity has a percent error of 1.6% while the upwind solution is off by 12.9%. Figure 4.5 represents the dimensionless U and V velocity contours, respectively, as a function of x/L at y/L = 0.5. Although there are no flagged grid points along the y/L = 0.5 line, the multigrid solution still shows an improvement over the upwind solution. This figure clearly shows the improvements to ϕ_0 outside the flagged region. For example, the U velocity profile at x/L = 0.654 has 7.0% and 11.2% errors for the MAD1-WFDS and upwind schemes, respectively.

Figures 4.6 and 4.7 show similar plots to Figures 4.3 and 4.4, but for the MAD2-WDS solution rather than MAD1-WFDS. Similarly, Figures 4.8 and 4.9 are the velocity profile results for MAD3-FDS. More specifically, Figs. 4.6 and 4.8 show the dimensionless velocity profiles at x/L = 0.5. Comparing the results of the V velocity profile at y/L = 0.731 (see Figs. 4.6 and 4.8), the percent error is 5.6% and 3.8% for the MAD2-WDS and MAD3-FDS schemes, respectively and 12.9% for the upwind solution. Figures 4.7 and 4.9 are the dimensionless velocity profiles at y/L = 0.5 for the MAD2-WDS and Mad3-FDS schemes, respectively. Again, note that the MAD

solutions show an improvement over the upwind solution although these profiles are not in the flagged region. At x/L = 0.654, the MAD2-WDS and MAD3-FDS solutions have percent errors of 8.0% and 7.6%, respectively, compared with 11.2% for the upwind solution.

The three multigrid solutions are compared to each other and the upwind, Burggraf and fine grid QUICK solutions in Figures 4.10 through 4.13. The three MAD schemes all result in improved solutions as compared to the fine grid (79 x 79) QUICK solution. In general, MAD1-WFDS gives the better solution of the three methods, but it requires more cpu effort to achieve the improvement, as evidenced in Table 4.1. This table gives the virtual and total cpu times of the three MAD, upwind and QUICK schemes on the 41 x 41 grid and the 79 x 79 QUICK solution. MAD2-WDS uses the least cpu time of the MAD schemes, but also shows the least improvement over the upwind solution, especially outside the flagged region. There is a direct correlation between the cpu effort required and the accuracy of the improved solution. If the cpu effort required per iteration is used as the judging criteria, however, then the MAD3-FDS scheme is the most cost effective (0.2869 cpu seconds/iteration) while the QUICK method is the most expensive (0.4837 seconds/iteration).

The difference in results between the three MAD methods may be accounted for in the following manner. In MAD2-WDS the solution on the flagged and unflagged regions are linked by

TIMING INFORMATION FOR FLOW IN A DRIVEN CAVITY

GRID SIZE AND SOLUTION SCHEME	VIRTUAL CPU (SEC)	TOTAL CPU (SEC)	No. of Cycles	NO. OF ITER	cpu/ ITER
41X41 grid, Upwind	140.99	153.58	---	450	0.3413
QUICK	227.55	241.84	---	500	0.4837
MAD1-WFDS	203.96	224.13	3	690	0.3244
MAD2-WDS	165.25	179.97	200	500	0.3599
MAD3-FDS	180.57	200.82	200	700	0.2869
79x79 grid, QUICK	2627.69	2732.05	---	1500	1.8214

Table 4.1 Computer timing information for flow in a driven cavity.

the influence of the neighbor grid across the subdomain boundary. Depending upon the flow direction, the unflagged upwind neighbor may have the dominating influence on the flagged QUICK neighbor. The influence of the improved QUICK solution in the subdomain on the unflagged global domain is restricted to its influence as a neighbor grid along the interface boundaries. In MAD1-WFDS, however, after a region is flagged the QUICK solution is obtained and then remains unchanged when its influence is spread in to the global domain via its use as a neighbor grid (or as a boundary value in MAD3-FDS). The resulting **improved** upwind value is then used as the new boundary value for the flagged subregions when the next round of the multigrid calculations begins.

4.6.2 Flow Over a Backward Facing Step

This problem is also widely used as a test case (see for example Armaly et al, 1983 and Moukalled, 1987). This problem is depicted in Fig. 4.14. The channel height h is 10.1 mm, while the step height s is 4.9 mm. The length examined is equvalent to 30s. The working fluid is air flowing with a Reynolds number, Re = 389. The governing equations are:

$$\frac{\partial}{\partial x}(\rho u^2) + \frac{\partial}{\partial y}(\rho uv) = -\frac{\partial p}{\partial x} + \mu\left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right] \qquad 4.25$$

and

$$\frac{\partial}{\partial x}(\rho uv) + \frac{\partial}{\partial y}(\rho v^2) = -\frac{\partial p}{\partial y} + \mu\left[\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right] \qquad 4.26$$

$$\frac{\partial}{\partial x}(\rho u) + \frac{\partial}{\partial y}(\rho v) = 0 \qquad 4.23$$

$$\frac{\partial}{\partial x}(\rho uT) + \frac{\partial}{\partial y}(\rho vT) = \frac{k}{c_p}\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) \qquad 4.24$$

where ρ is the density, μ is the dynamic viscosity and u and v are the velocities in the x- and y-directions, respectively. The dimensionless temperature is defined as follows:

$$T = \frac{t - t_{inlet}}{t_{wall} - t_{inlet}}$$

The boundary conditions are as follows:

$$u_{inlet} = 0 \qquad \text{for } y \le s$$

$$u_{inlet} = u_{max}\left[1 - \frac{\left(y - \frac{y+s}{2}\right)^2}{\left(s - \frac{y+s}{2}\right)^2}\right] \qquad \text{for } s \le y \le \frac{h+s}{2}$$

$$u_{inlet} = u_{max}\left[1 - \frac{\left(y - \frac{y+s}{2}\right)^2}{\left(h - \frac{y+s}{2}\right)^2}\right] \qquad \text{for } \frac{h+s}{2} \le y \le h$$

$$u = v = 0 \qquad \text{at the walls}$$

$$v = 0 \qquad \text{at the inlet}$$

$$\frac{\partial u}{\partial x}, \frac{\partial v}{\partial y} = 0 \qquad \text{at the outflow boundary}$$

where $u_{max} = 0.846$ m/s.

$T = 0$ at the inlet,

$T = 1$ at the bottom wall,

$$\frac{\partial T}{\partial x} = 0 \qquad \text{at the outflow boundary}$$

$$q'' = -k\frac{\partial T}{\partial y} = 0 \qquad \text{at the insulated top wall}$$

The problem was solved on a 47 x 38 grid. A QUICK solution on a 62 x 50 grid was used as an "exact" solution. The results are also compared with the experimental results obtained by Armaly, et al (1983).

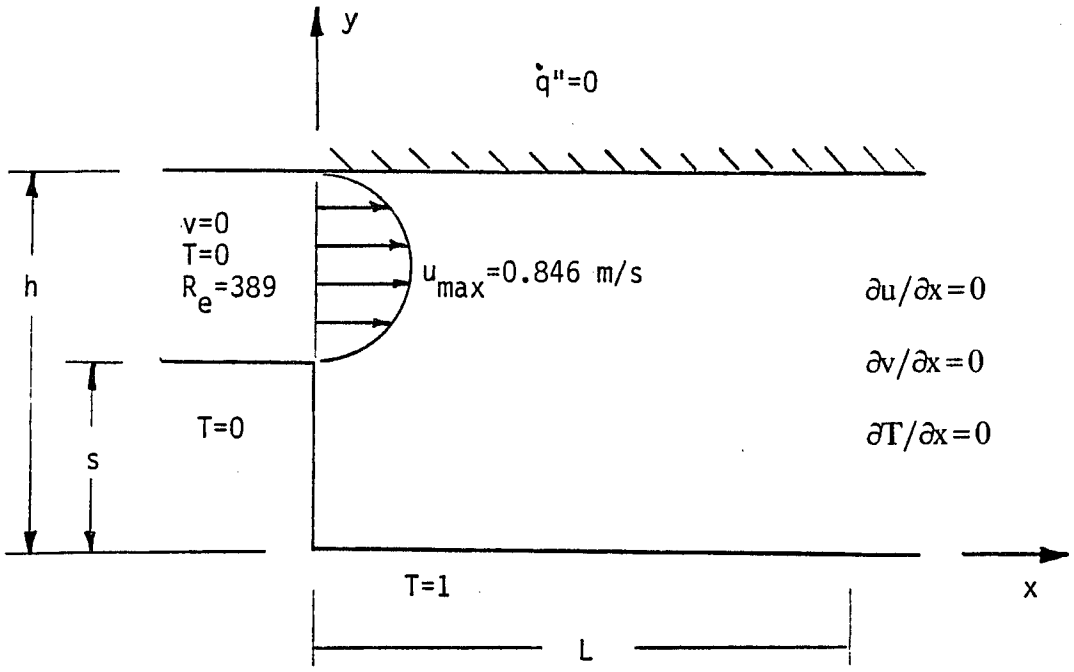


Fig. 4.14 Physical domain for flow over a backward facing step.

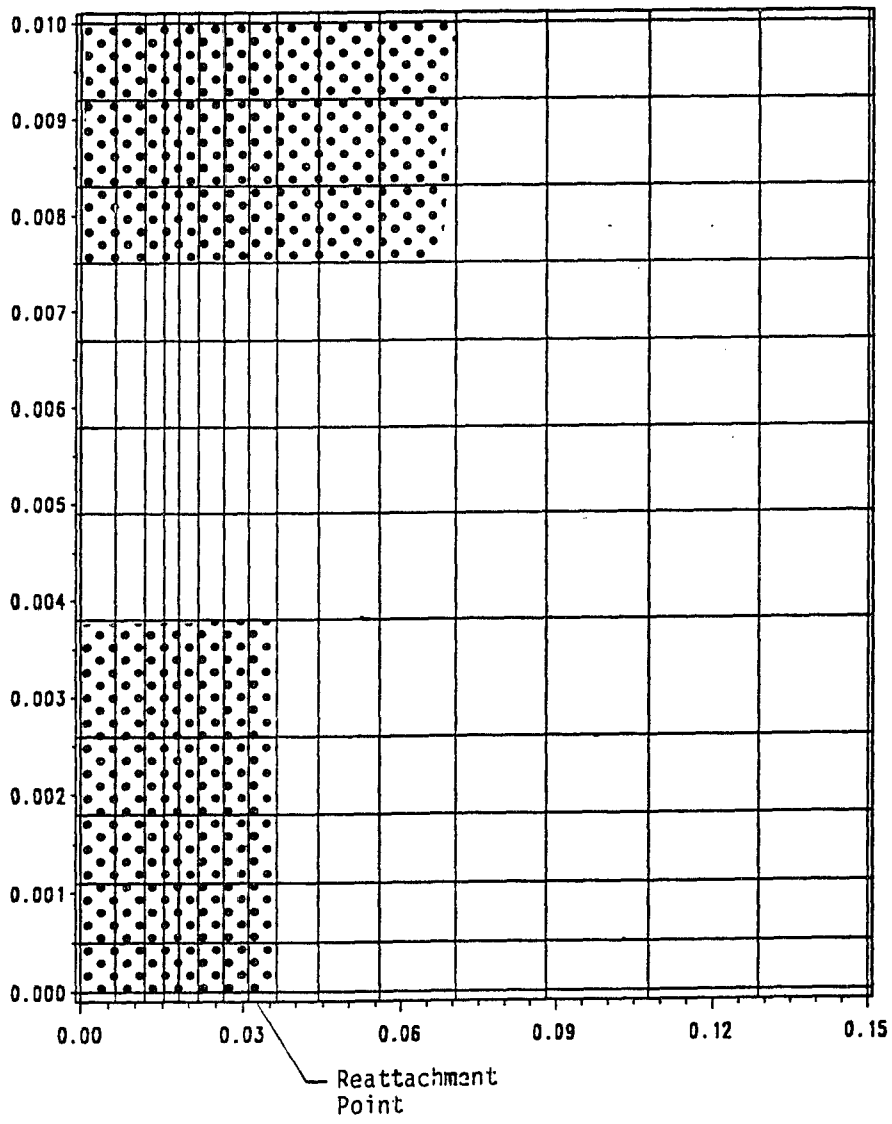


Fig. 4.15 Discretized domain and flagged regions for the velocity field of the flow over a backward facing step problem.

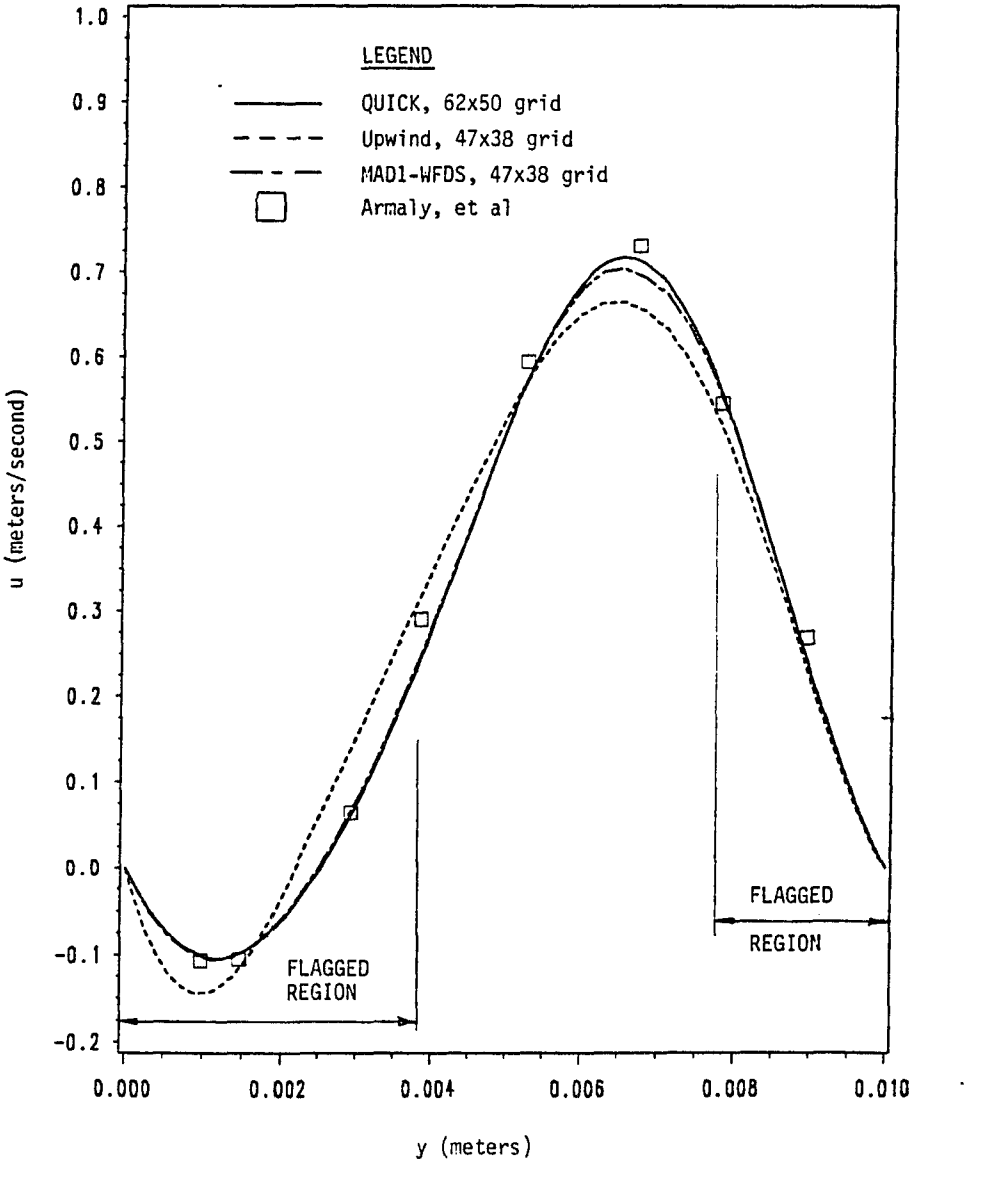


Fig. 4.16 u-velocity profile at x/s = 4.18 for flow over a backward facing step problem. Comparing upwind, MAD1-WFDS, Armaly, et al and fine grid QUICK solutions.

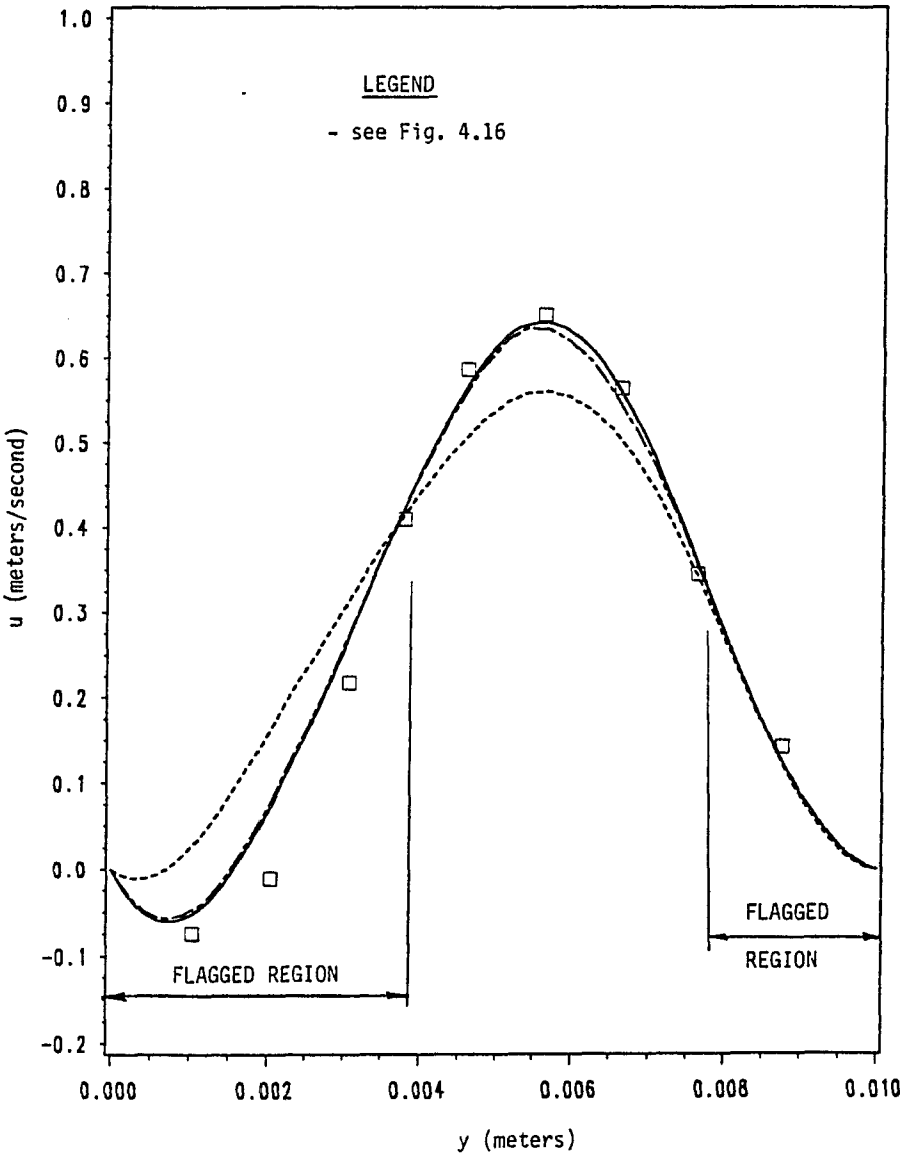


Fig. 4.17 u-velocity profile at x/s = 6.12 for flow over a backward facing step problem. Comparing upwind, MAD1-WFDS, Armaly, et al and fine grid QUICK solutions.

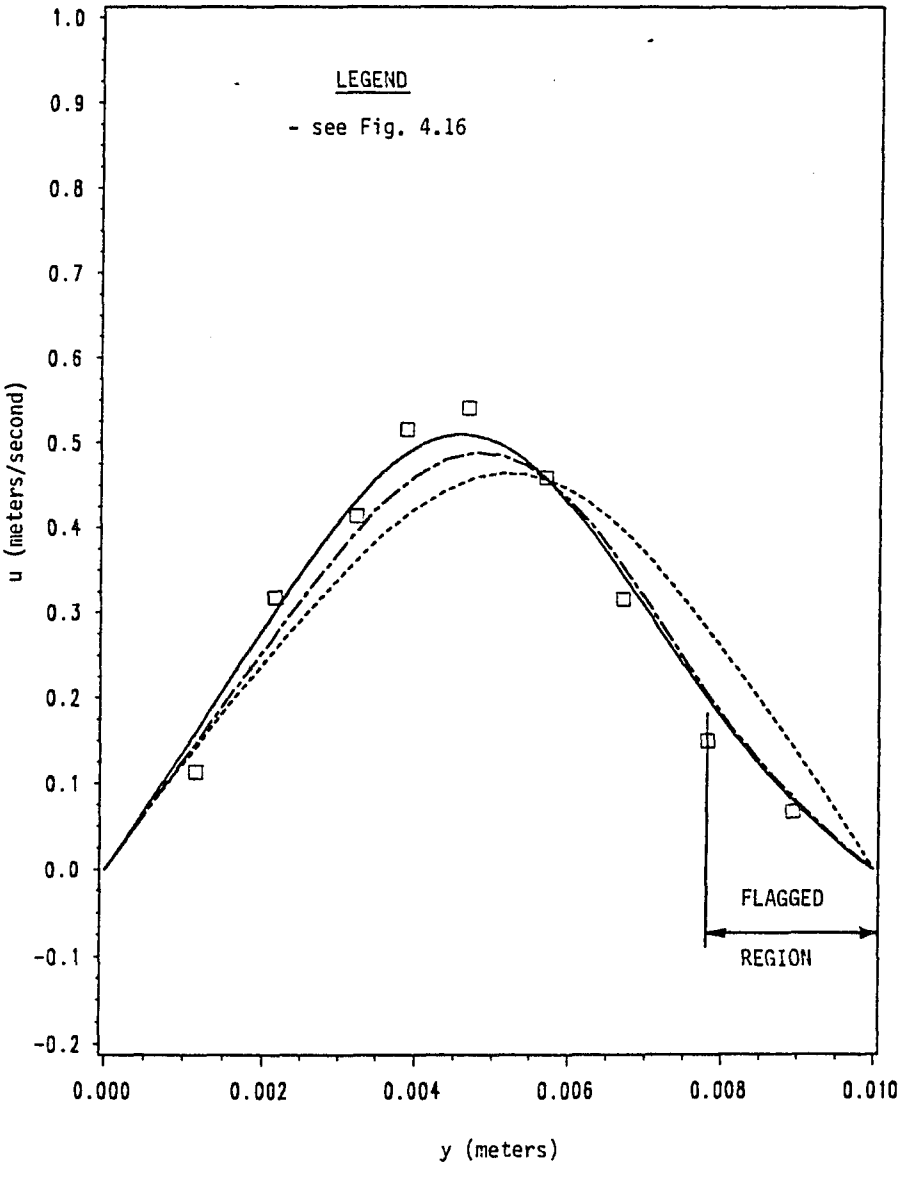


Fig. 4.18 u-velocity profile at x/s = 11.07 for flow over a backward facing step problem. Comparing upwind, MAD1-WFDS, Armaly, et al and fine grid QUICK solutions.

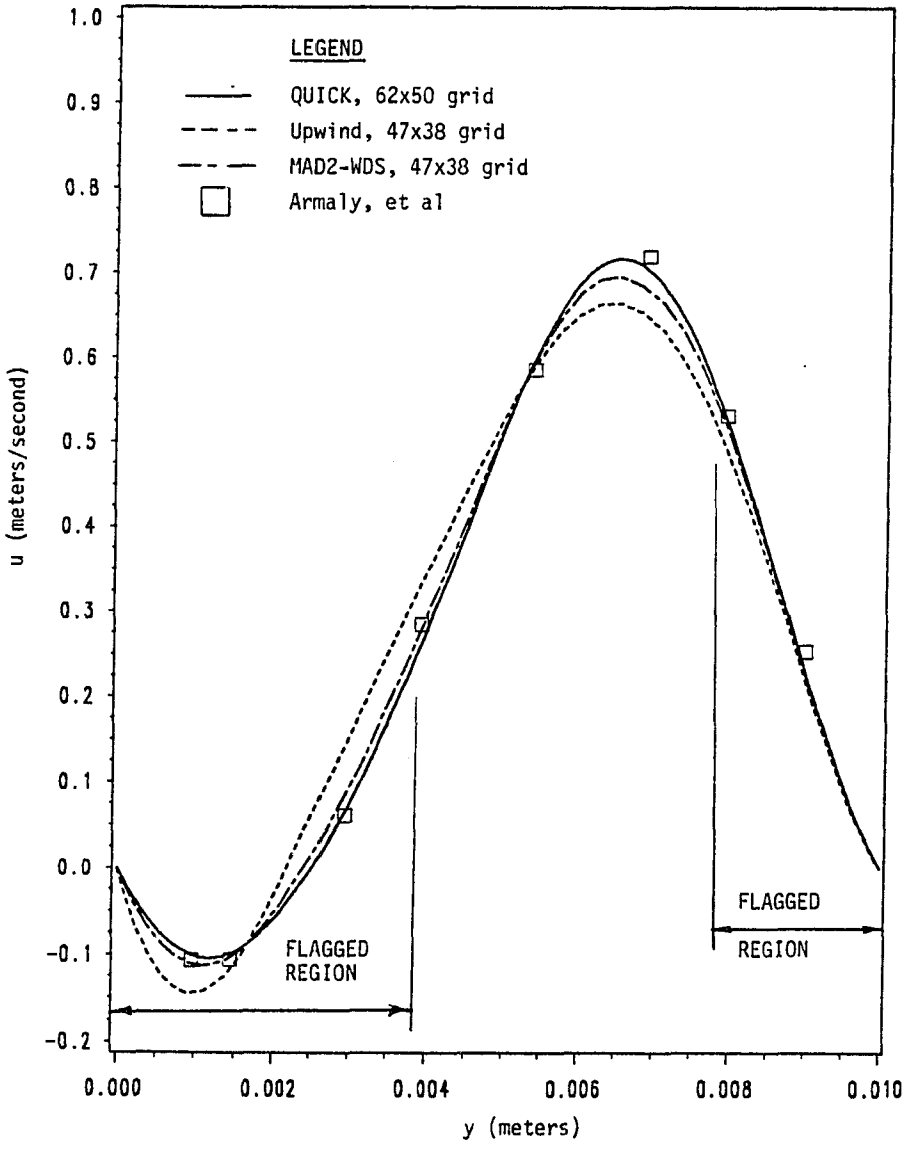


Fig. 4.19 u-velocity profile at x/s = 4.18 for flow over a backward facing step problem. Comparing upwind, MAD2-WDS, Armaly, et al and fine grid QUICK solutions.

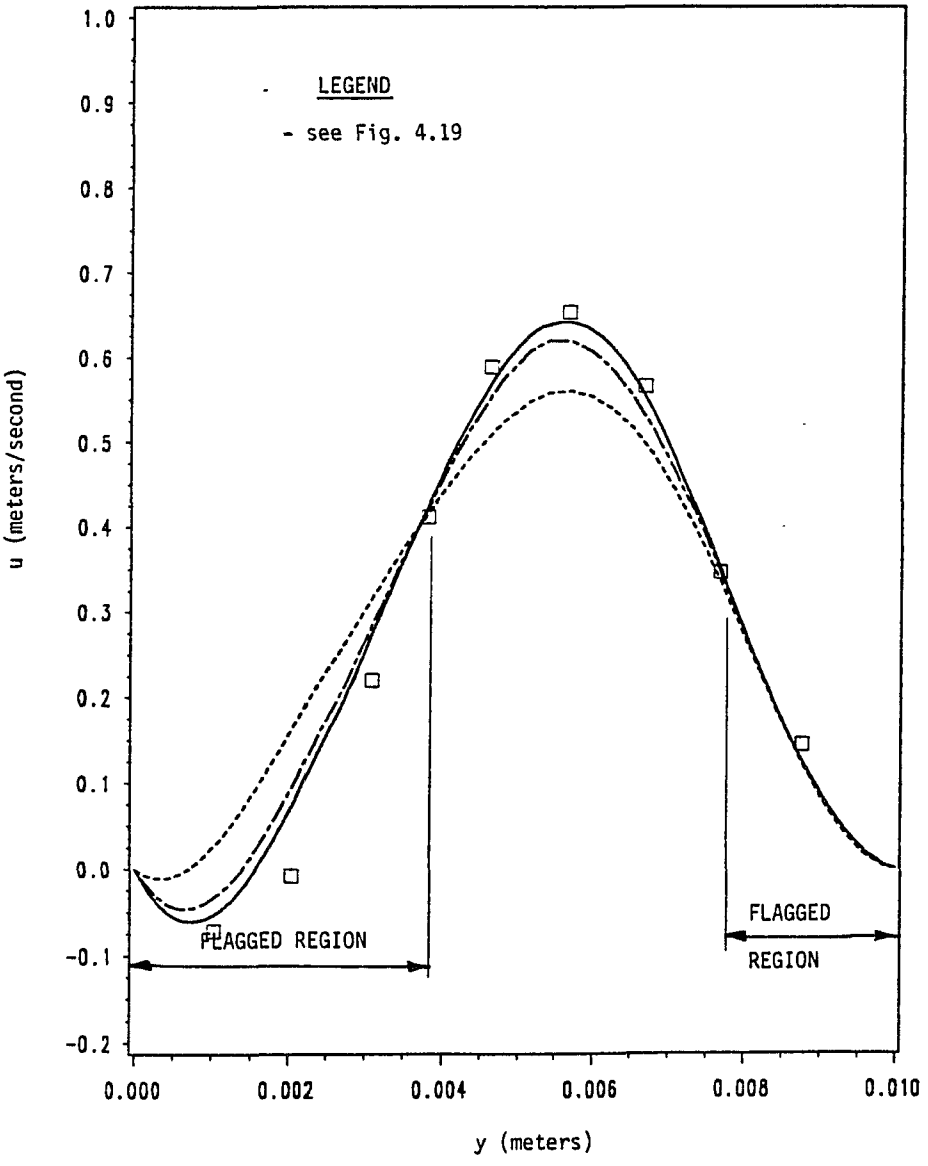


Fig. 4.20 u-velocity profile at x/s = 6.12 for flow over a backward facing step problem. Comparing upwind, MAD2-WDS, Armaly, et al and fine grid QUICK solutions.

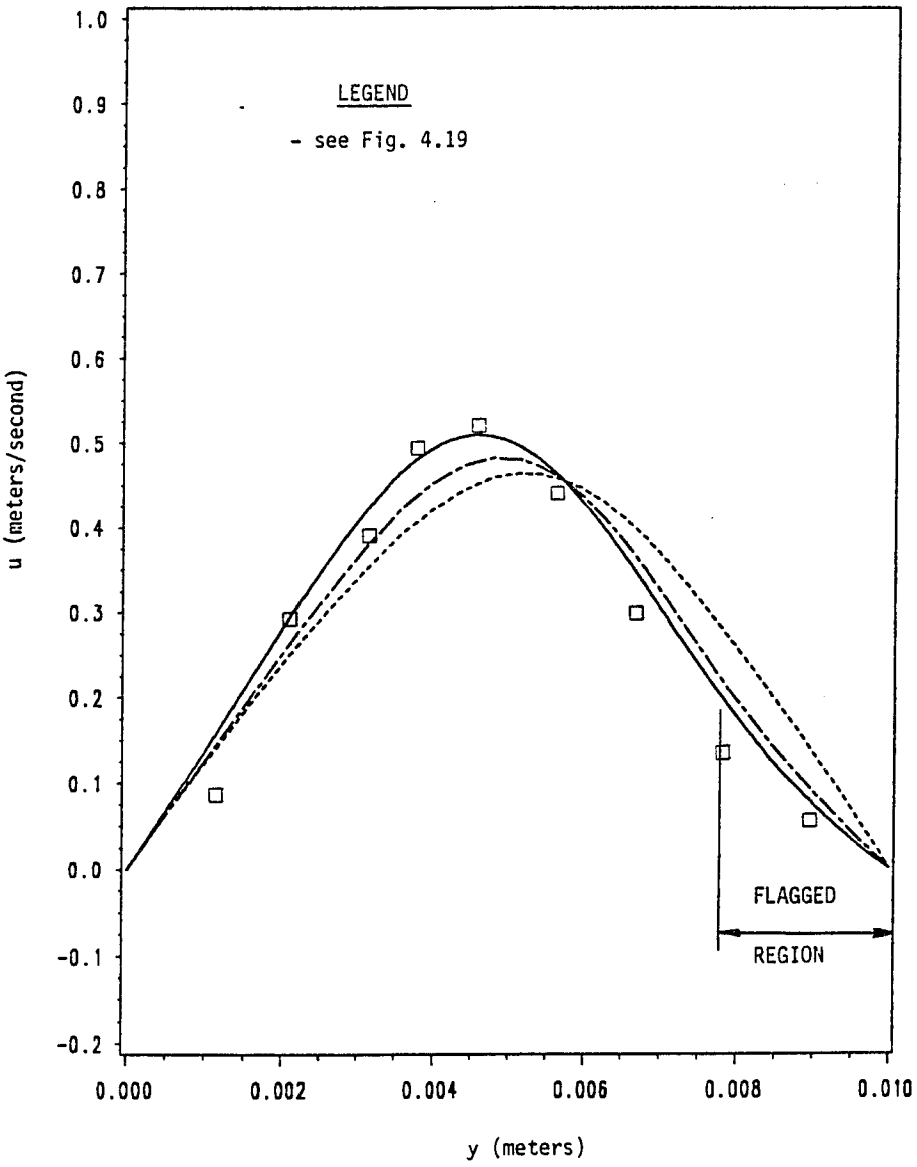


Fig. 4.21 u-velocity profile at x/s = 11.07 for flow over a backward facing step problem. Comparing upwind, MAD2-WDS, Armaly, et al and fine grid QUICK solutions.

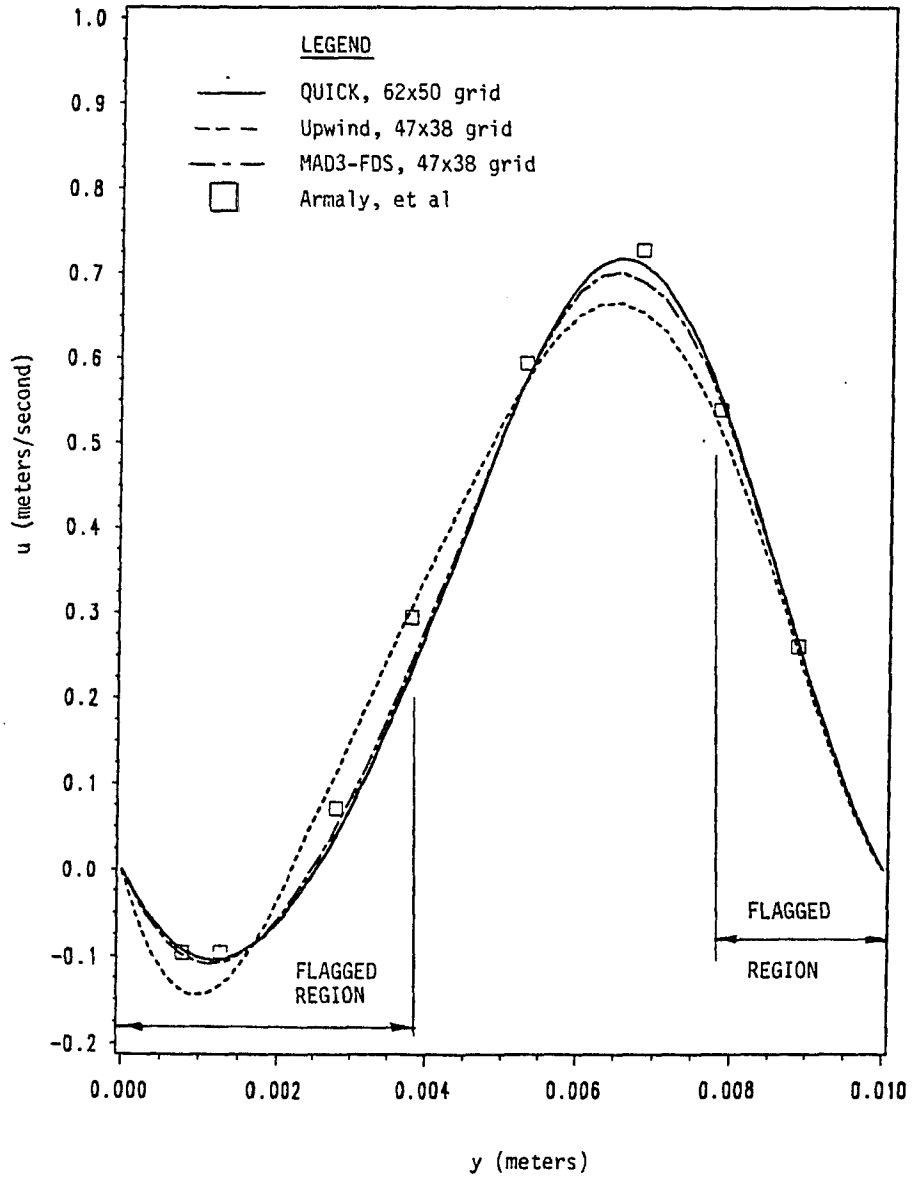


Fig. 4.22 u-velocity profile at x/s = 4.18 for flow over a backward facing step problem. Comparing upwind, MAD3-FDS, Armaly, et al and fine grid QUICK solutions.

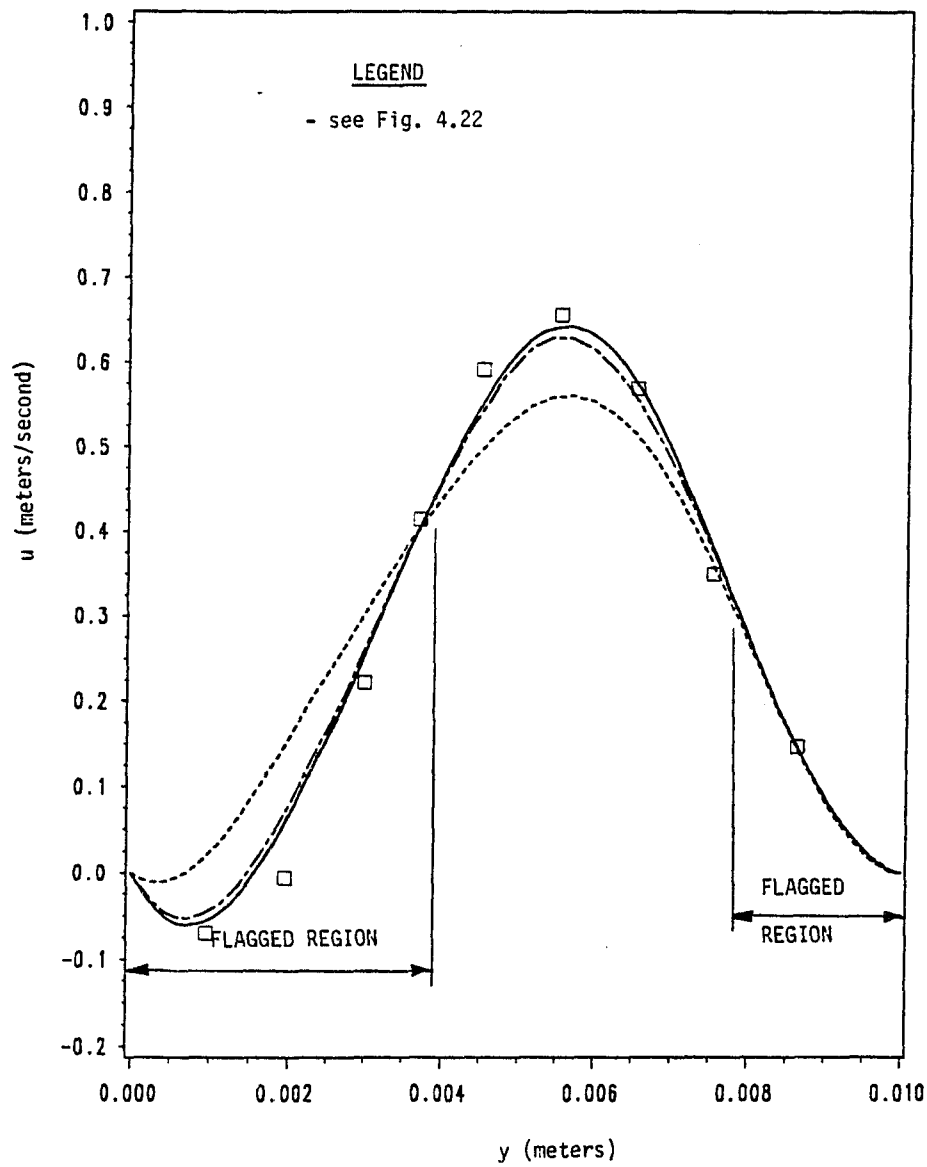


Fig. 4.23 u-velocity profile at x/s = 6.12 for flow over a backward facing step problem. Comparing upwind, MAD3-FDS, Armaly, et al and fine grid QUICK solutions.

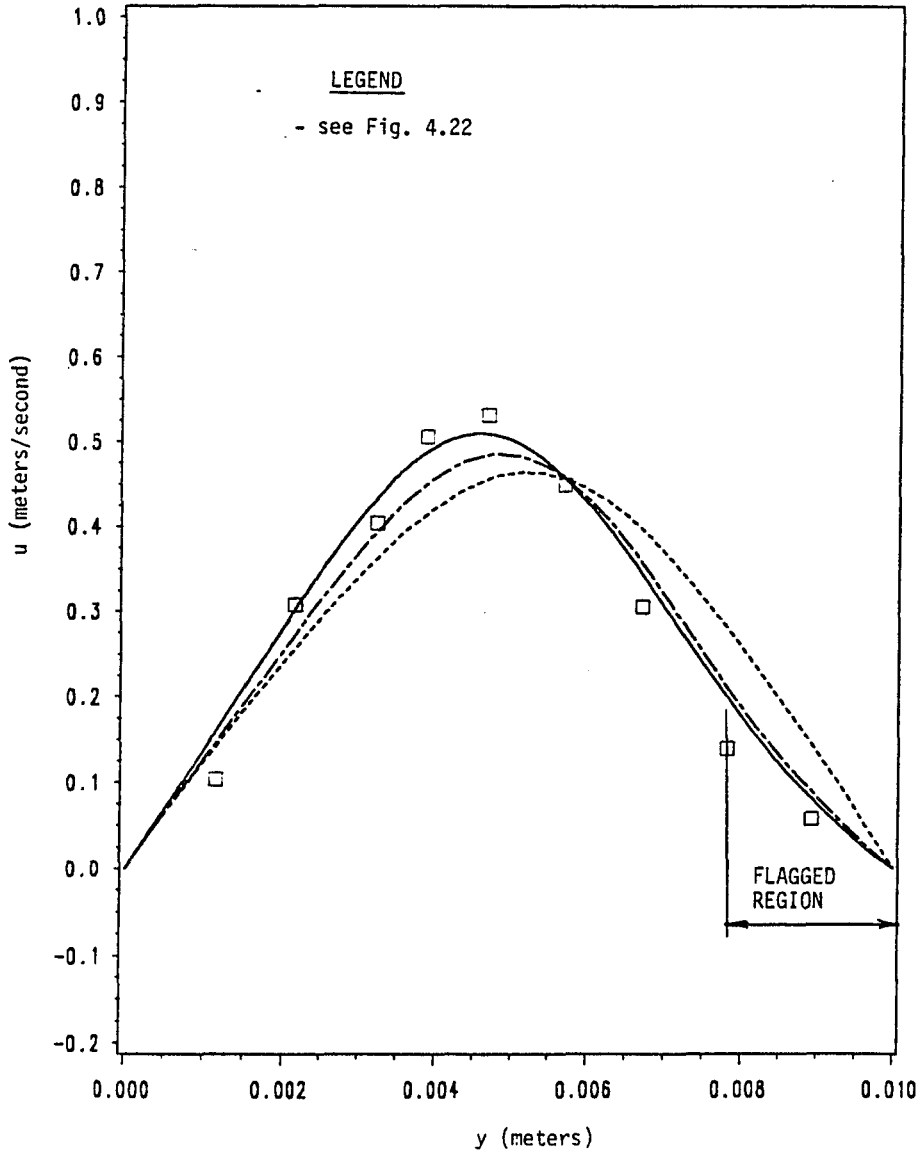


Fig. 4.24 u-velocity profile at x/s = 11.07 for flow over a backward facing step problem. Comparing upwind, MAD3-FDS, Armaly, et al and fine grid QUICK solutions.

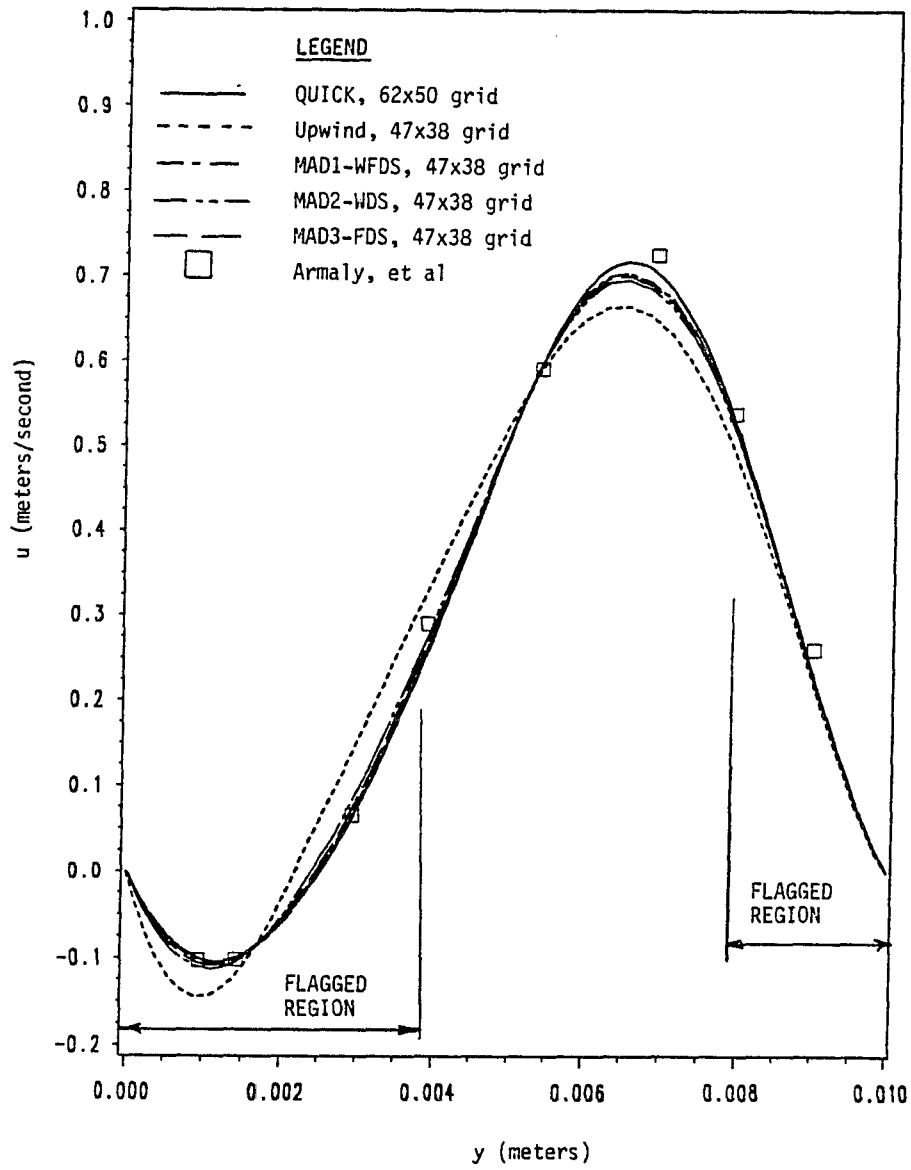


Fig. 4.25 u-velocity profile at x/s = 4.18 for flow over a backward facing step problem. Comparing upwind, MAD1-WFDS, MAD2-WDS, MAD3-FDS, and Armaly, et al with the fine grid QUICK solution.

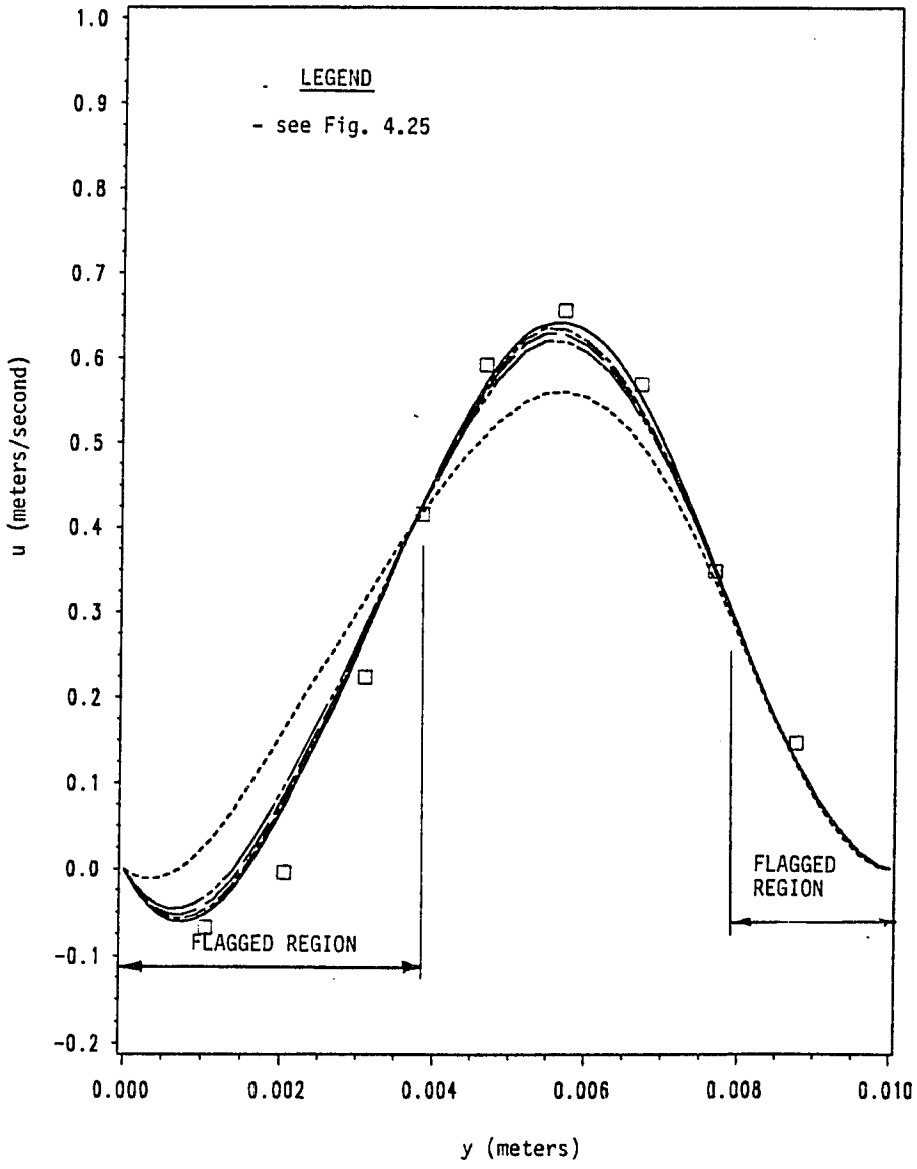


Fig. 4.26 u-velocity profile at x/s = 6.12 for flow over a backward facing step problem. Comparing upwind, MAD1-WFDS, MAD2-WDS, MAD3-FDS, Armaly, et al and the fine grid QUICK solutions.

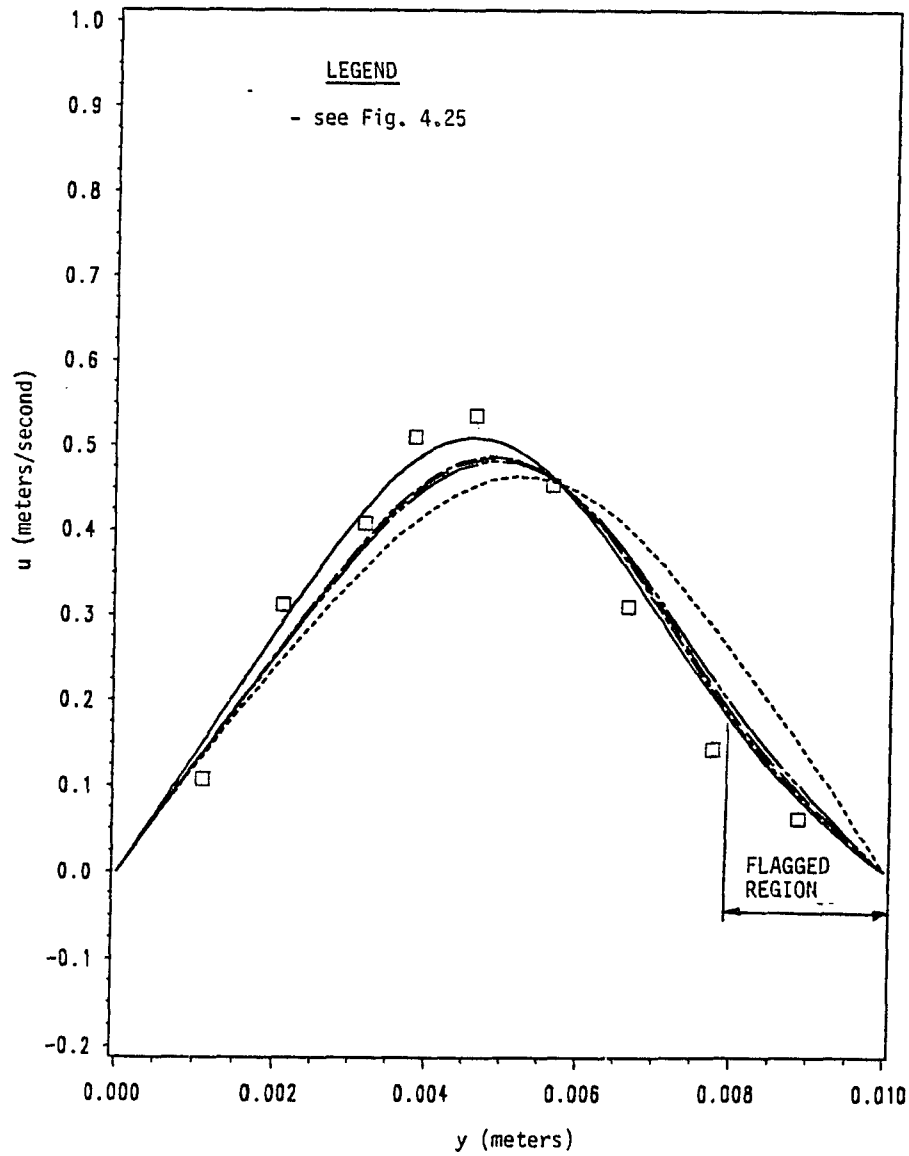


Fig. 4.27 u-velocity profile at x/s = 11.07 for flow over a backward facing step problem. Comparing upwind, MAD1-WFDS, MAD2-WDS, MAD3-FDS, Armaly, et al and the fine grid QUICK solutions.

TIMING INFORMATION FOR FLOW OVER A BACKWARD FACING STEP

GRID SIZE AND SOLUTION SCHEME	VIRTUAL CPU (SEC)	TOTAL CPU (SEC)	NO. OF CYCLES	NO. OF ITER	cpu/ ITER
47x38 grid, Upwind	329.85	347.22	---	1100	0.3157
QUICK	688.44	719.23	---	1500	0.4795
MAD1-WFDS	437.34	480.68	4/3	1450	0.3315
MAD2-WDS	334.65	354.68	250/200	1100	0.3224
MAD3-FDS	397.46	432.26	250/150	1500	0.2882
62x50 grid, QUICK	2581.21	2738.82	---	3000	0.9129

Note: The two numbers separated by a / in the "NO. OF CYCLES" column represent the number of cycles for the velocity and temperature fields, respectively.

Table 4.2 Computer timing information for flow over a backward facing step problem.

Figure 4.15 shows the discretized domain and flagged regions for the velocity field on a 17 x 14 grid. The flagged region is proportionately identical to that of the 47 x 38 grid. Again, the smaller grid size is used in Fig. 4.15 for clarity.

Figures 4.16, 4.17 and 4.18 are the u-velocity profiles as a function of y at three different x/s locations. The upwind and MAD1-WFDS solutions are compared with the fine grid QUICK solution and the experimental results of Armaly, et al (1983). The percentage error can be defined as

$$\% \text{ error} = 100 \times | \text{QUICK} - Z | / \text{QUICK}$$

where QUICK is the QUICK solution and Z is the upwind or one or the MAD schemes. As in the flow in a driven cavity problem, the MAD methods yield solutions which are more accurate than the upwind method, even outside the flagged regions. The maximum percent errors outside of the flagged regions for the upwind method are 7.3%, 13.7% and 11.8% at x/s = 4.18, 6.12 and 11.07, respectively. For MAD1-WFDS, however, the maximum percent errors at the same locations are 1.4%, 2.54% and 5.4%.

Figures 4.19, 4.20 and 4.21 are similar to plots shown in Figs. 4.16 - 4.18, except that the MAD2-WDS solution (rather than MAD1-WFDS) is compared with the upwind, Armaly, et al and fine grid QUICK solutions. Figures 4.22 - 4.24 are the velocity profiles for MAD3-FDS at the same x/s locations used in presenting the results for methods MAD1-WFDS and MAD2-WDS. For these two MAD schemes, the solution is again an

improvement over the upwind scheme as compared with the fine grid (62 x 50) QUICK solution. Here also, the improvements extend outside the flagged regions as for the MAD1-WFDS method. At x/s = 4.18 (see Figs. 4.19 and 4.22), the maximum percent errors for the MAD2-WDS, MAD3-FDS and upwind schemes are 2.8%, 2.1%, and 7.13%, respectively. At x/s = 6.12 (see Figs. 4.20 and 4.23), the percent errors are 4.5% for MAD2-WDS, 2.9% for MAD3-FDS and 13.7% for the upwind methods. The maximum percent errors outside the flagged regions in Figs. 4.21 and 4.24 (at x/s = 11.07) are 6.9%, 5.9% and 11.8% for the MAD2-WDS, MAD3-FDS and upwind methods, respectively.

Figures 4.25 through 4.27 compare the solutions for the three MAD methods with the upwind and fine grid QUICK solutions and with the experimental results of Armaly, et al (1983) for the velocity field at the same three x/s locations. Again, the improvements to the overall solution effected by the multigrid adaptive differencing techniques are clearly a function of the cpu effort required.

Table 4.2 presents the timing information for this test case problem. MAD1-WFDS needs the most cpu time, but still only uses 63.5% of the cpu time required for the QUICK solution on the same grid size. MAD2-FDS uses the least time (334.65 sec) again, but also yields the least improvement over the upwind solution of the three MAD methods. MAD3-FDS requires the least cpu effort per iteration, 0.2882 seconds/iteration.

Figure 4.28 shows the flagged region for the temperature field on a 17 x 14 grid. Again, the flagged region is proportionately identical to the flagged region on the 47 x 38 grid.

Figures 4.29, 4.30 and 4.31 show the temperature field solutions generated by the upwind, MAD1-WFDS and fine grid QUICK solutions at three x/s locations (x/s = 2.55, x/s = 6.12 and x/s = 8.52). At each x/s location the MAD1-WFDS solution is much closer to the "exact" solution than the upwind results, even outside the flagged region. For example, in Fig. 4.31 (x/s = 8.52) at y = 0.0035, the maximum percent error (as defined above) is 11.8% for MAD1-WFDS and 62.8% for the upwind method.

Figures 4.32 through 4.37 present the temperature profile results for the MAD2-WDS and MAD3-FDS algorithms at the same three x/s locations. The results are compared with upwind and fine grid QUICK solutions and the experimental results of Armaly, et al. For both adaptive differencing schemes, the temperature profiles show significant improvement over the upwind solution, both in and outside of the flagged subdomain. Comparing the maximum percent error at the same location (i.e., at y = 0.0035 for the x/s = 8.52 temperature profile), the MAD2-WDS method (see Fig. 4.34) has a 17.6% error while the MAD3-FDS method (see Fig. 4.37) has a 5.9% error. The upwind percent error at this location is 62.8%, as stated earlier.

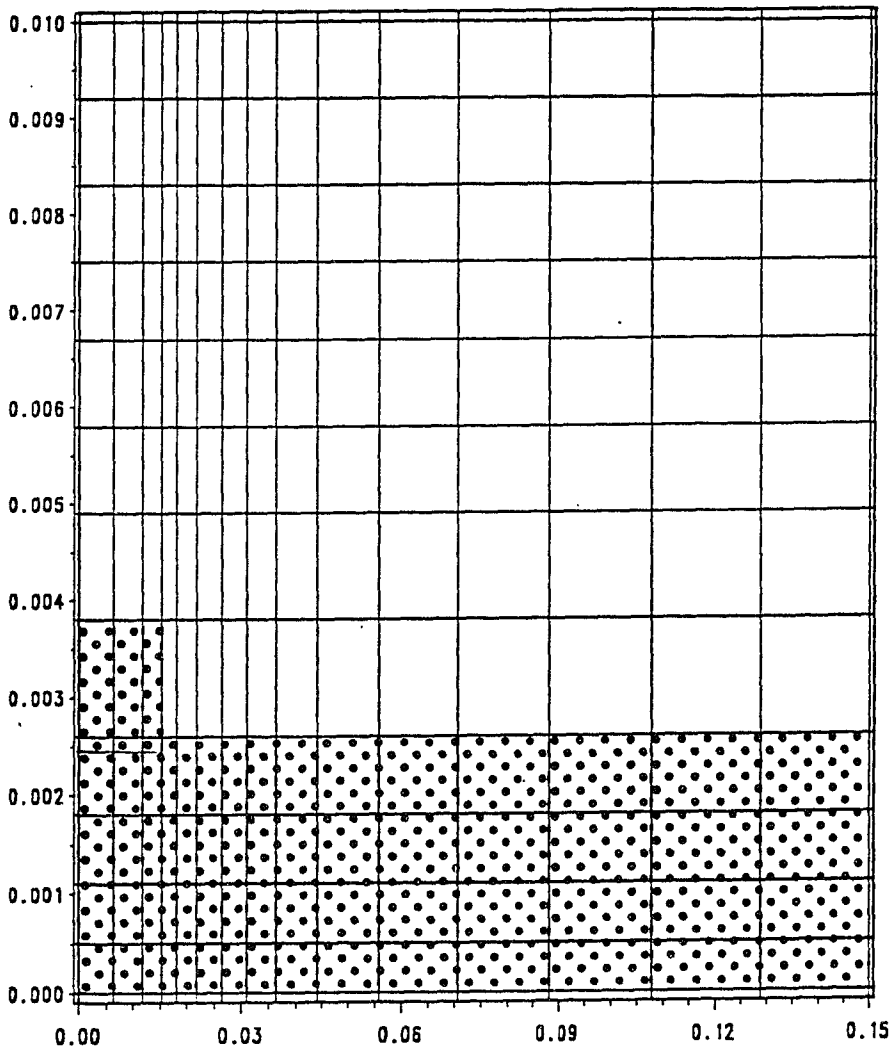


Fig. 4.28 Discretized domain and flagged region for the temperature field of the flow over a backward facing step problem.

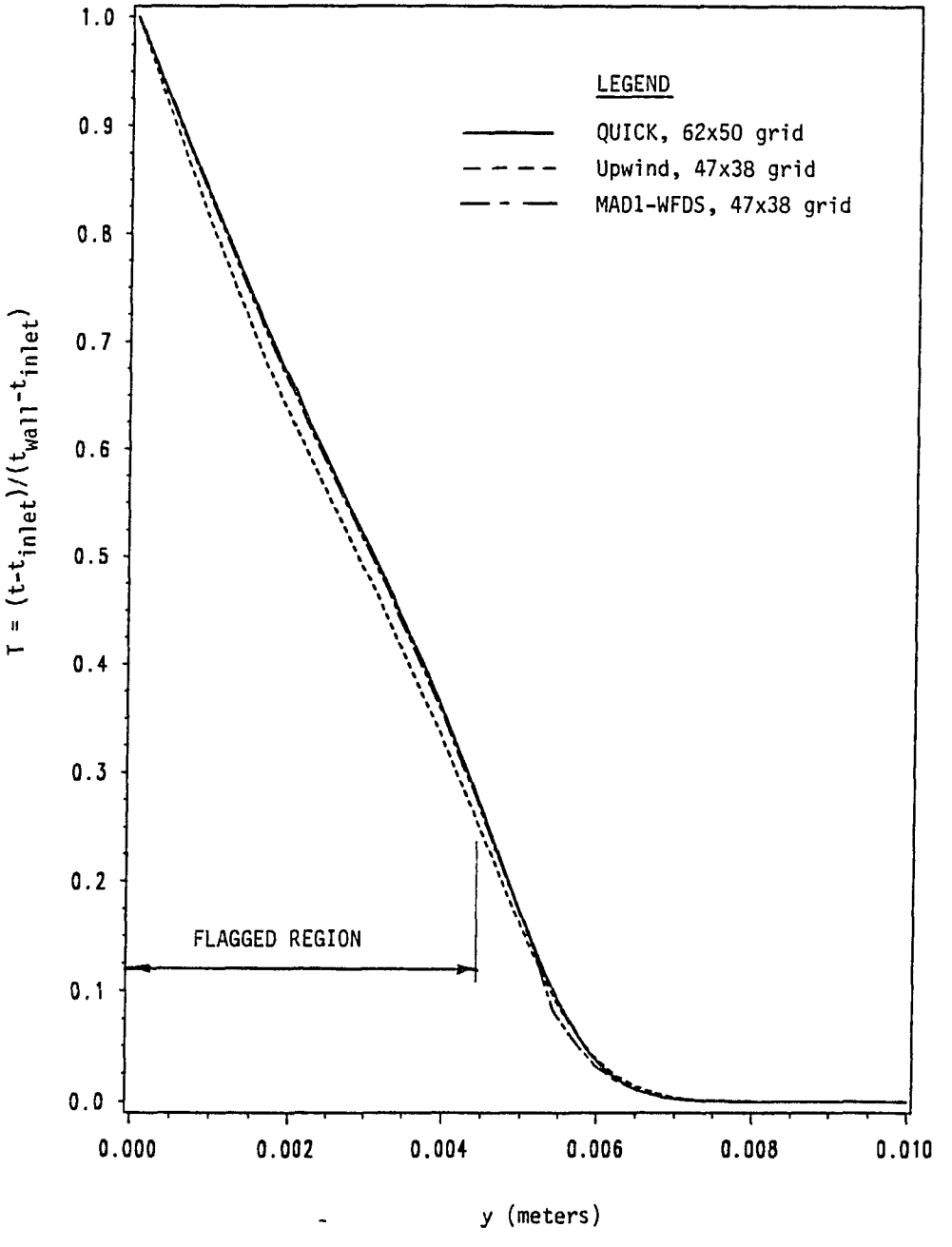


Fig. 4.29 Temperature profile at x/s = 2.55 for flow over a backward facing step problem. Comparing upwind, MAD1-WFDS and fine grid QUICK solutions.

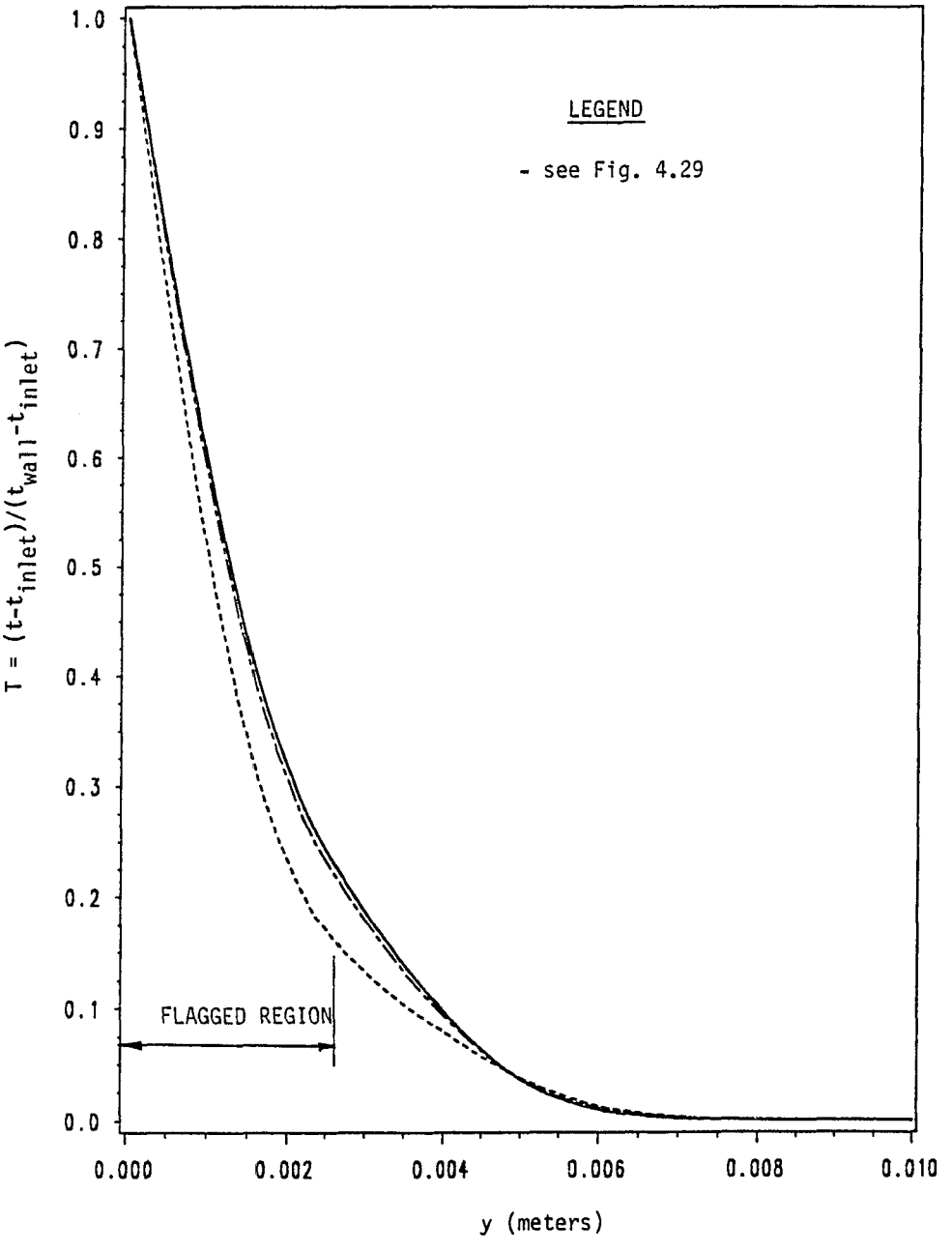


Fig. 4.30 Temperature profile at x/s = 6.12 for flow over a backward facing step problem. Comparing upwind, MAD1-WFDS and fine grid QUICK solutions.

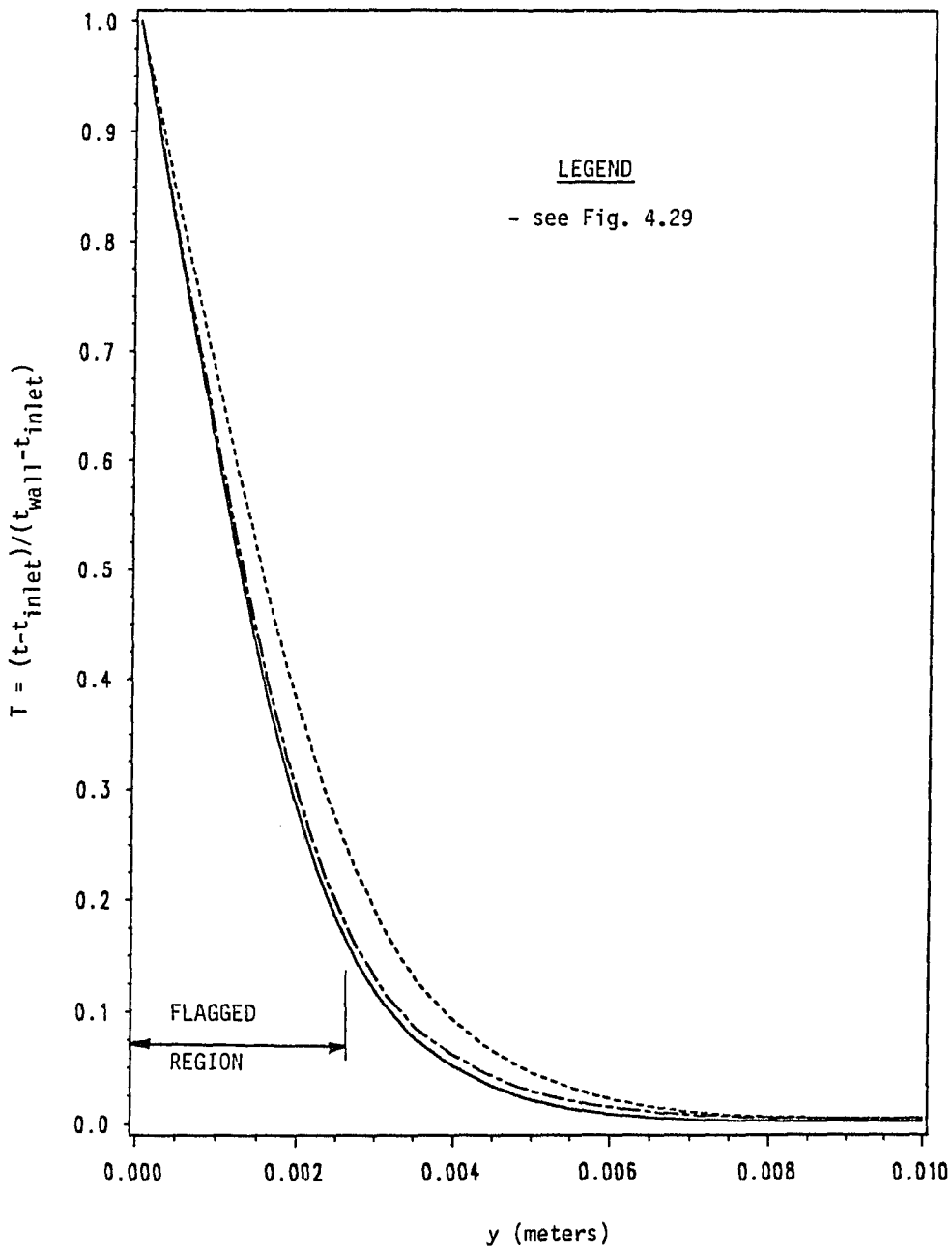


Fig. 4.31 Temperature profile at x/s = 8.52 for flow over a backward facing step problem. Comparing upwind, MAD1-WFDS and fine grid QUICK solutions.

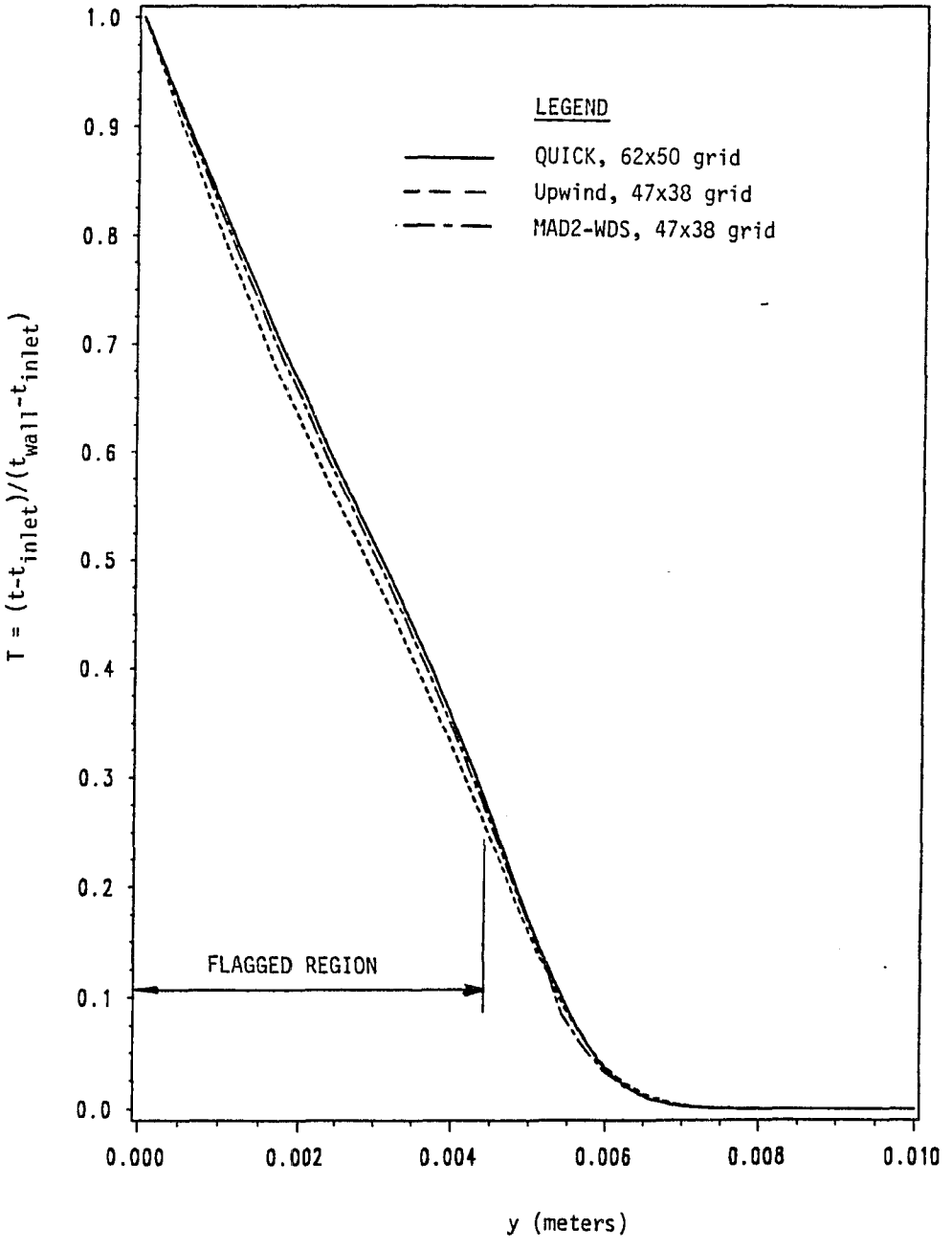


Fig. 4.32 Temperature profile at x/s = 2.55 for flow over a backward facing step problem. Comparing upwind, MAD2-WDS and fine grid QUICK solutions.

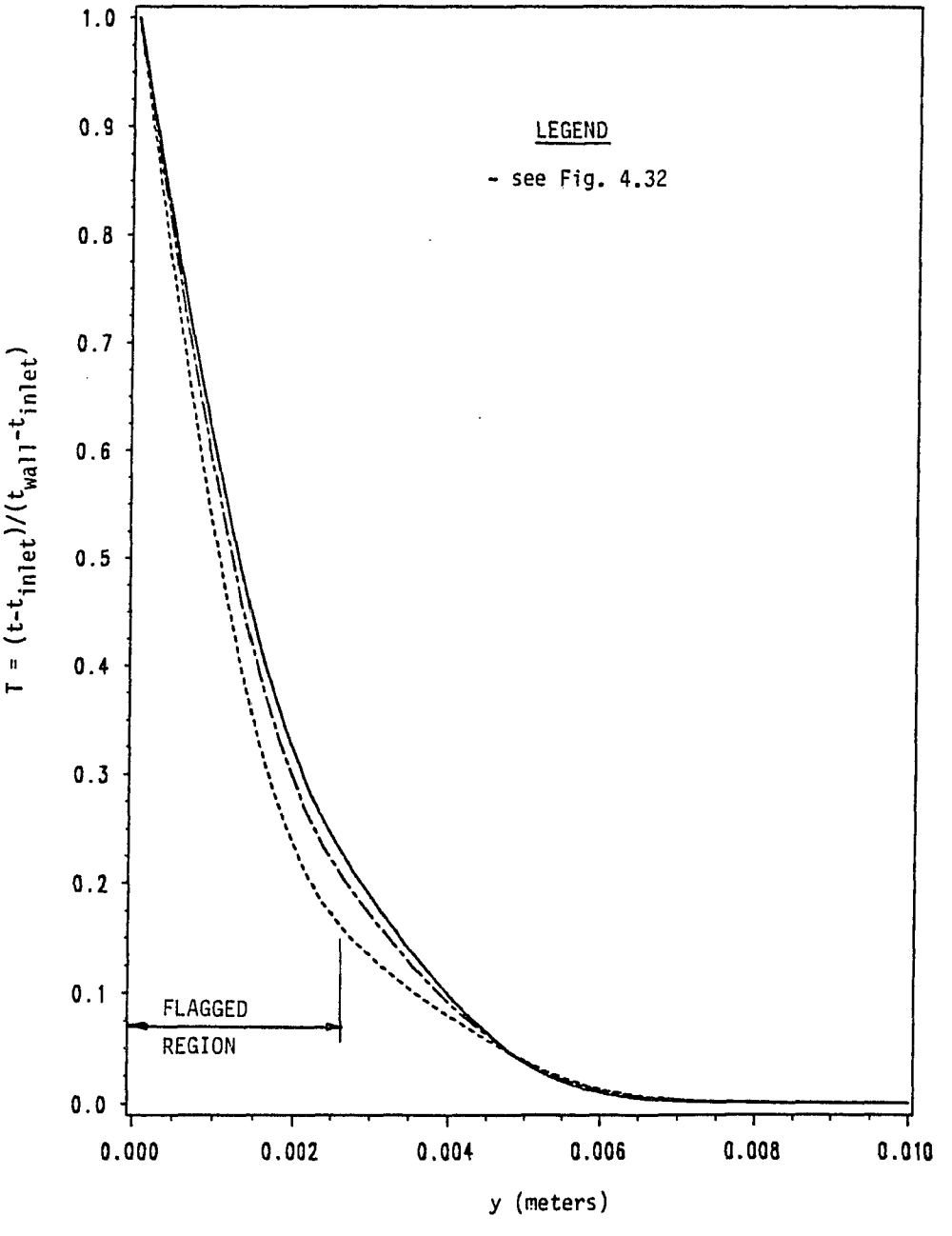


Fig. 4.33 Temperature profile at x/s = 6.12 for flow over a backward facing step problem. Comparing upwind, MAD2-WDS and fine grid QUICK solutions.

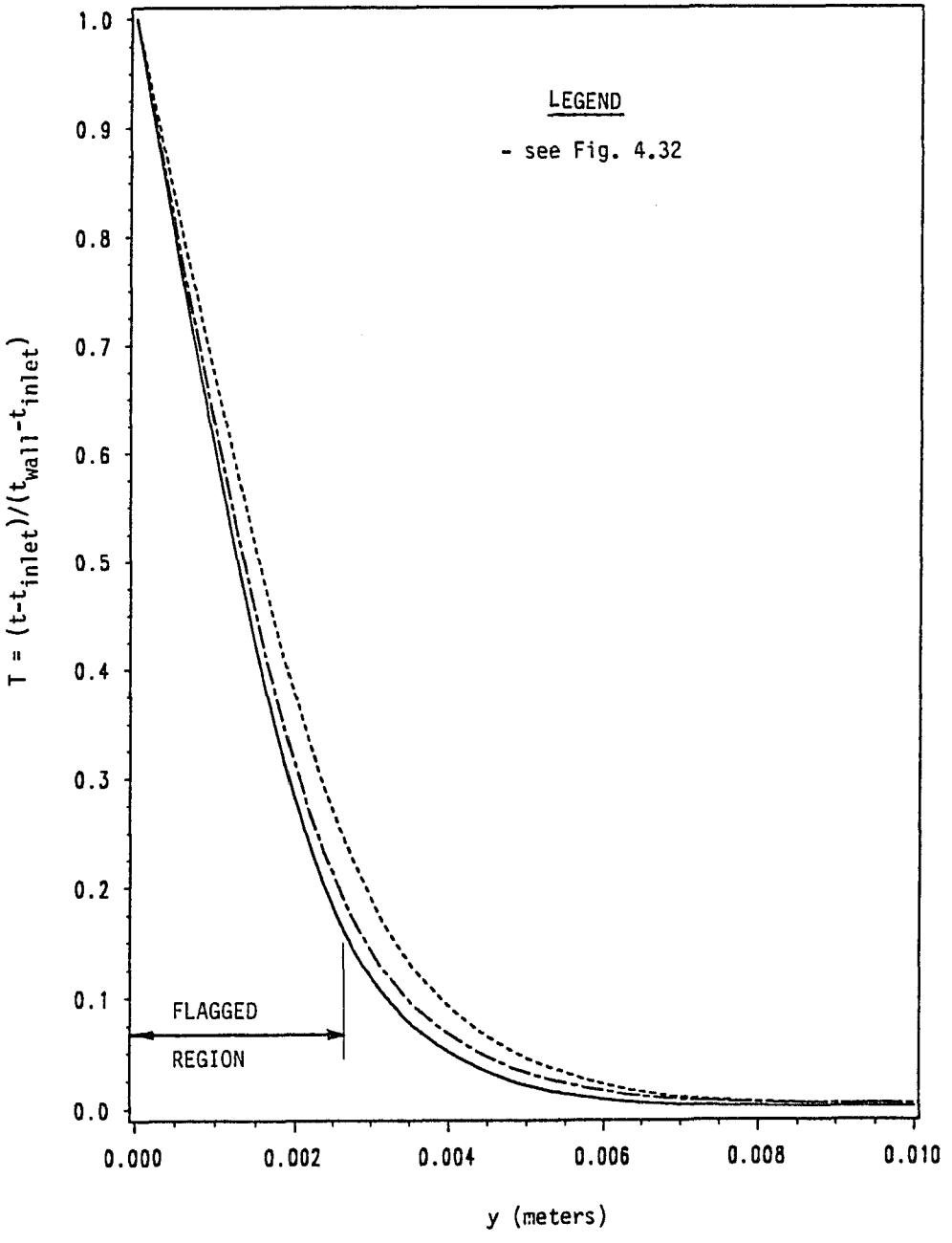


Fig. 4.34 Temperature profile at x/s = 8.52 for flow over a backward facing step problem. Comparing upwind, MAD2-WDS and fine grid QUICK solutions.

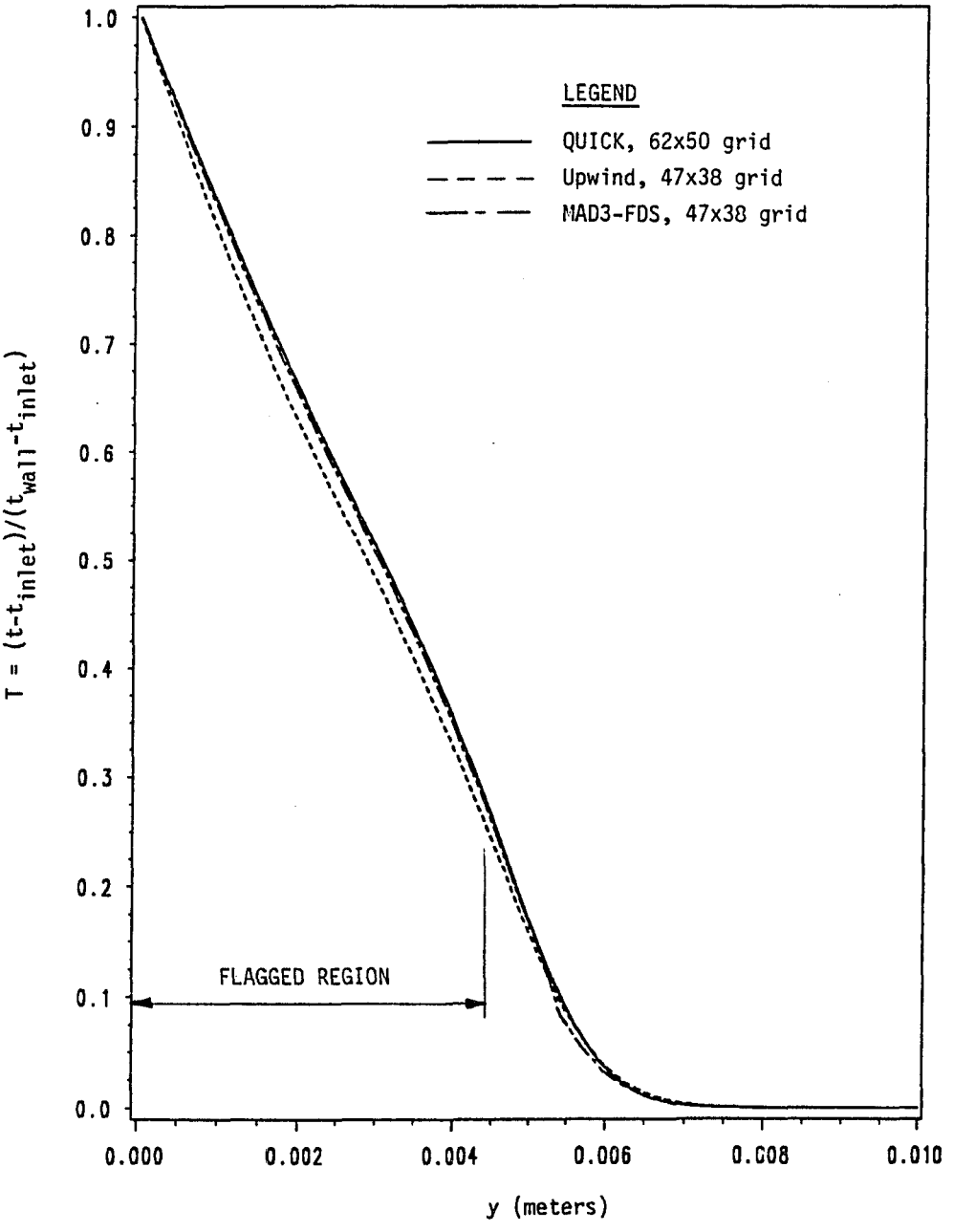


Fig. 4.35 Temperature profile at x/s = 2.55 for flow over a backward facing step problem. Comparing upwind, MAD3-FDS and fine grid QUICK solutions.

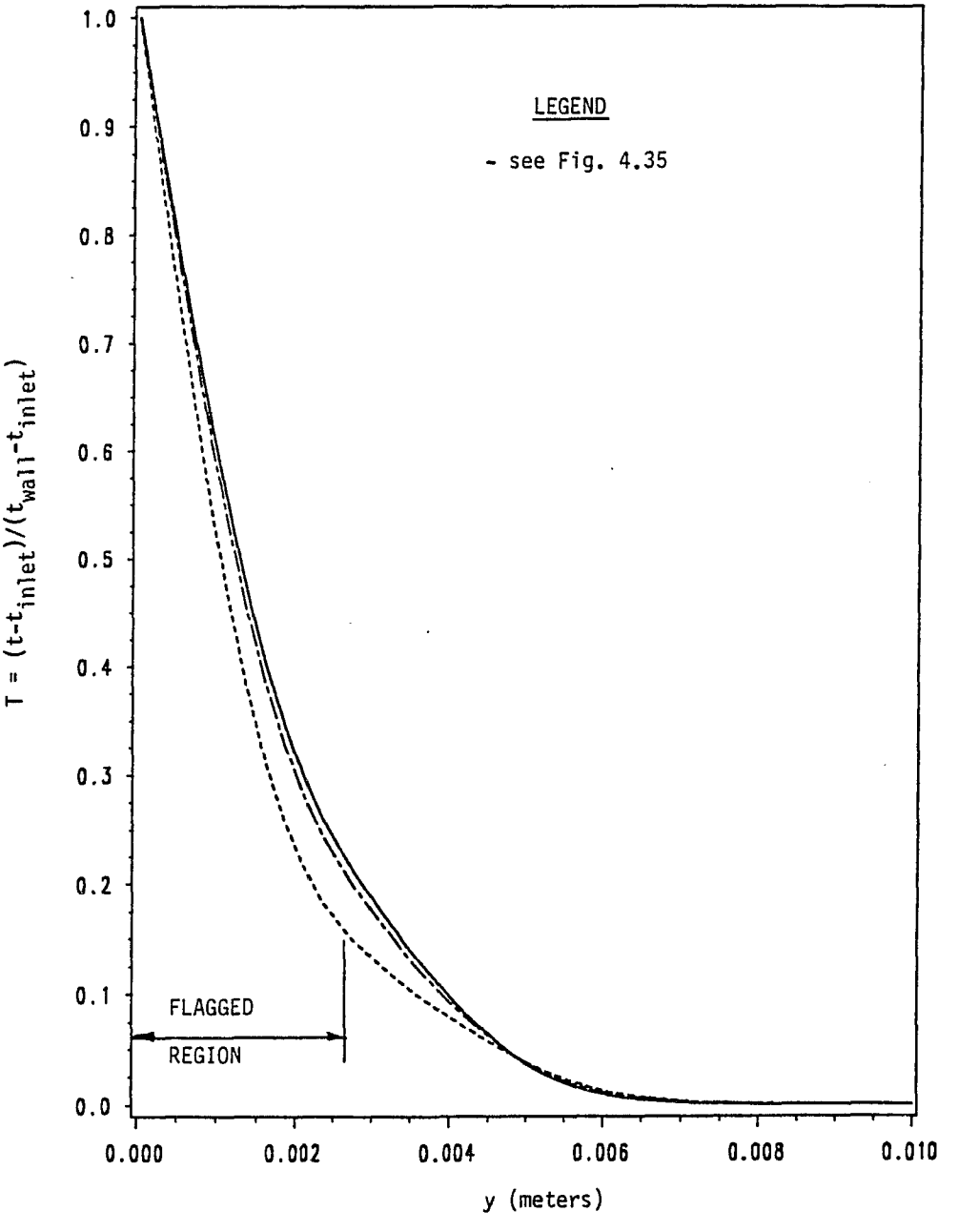


Fig. 4.36 Temperature profile at x/s = 6.12 for flow over a backward facing step problem. Comparing upwind, MAD3-FDS and fine grid QUICK solutions.

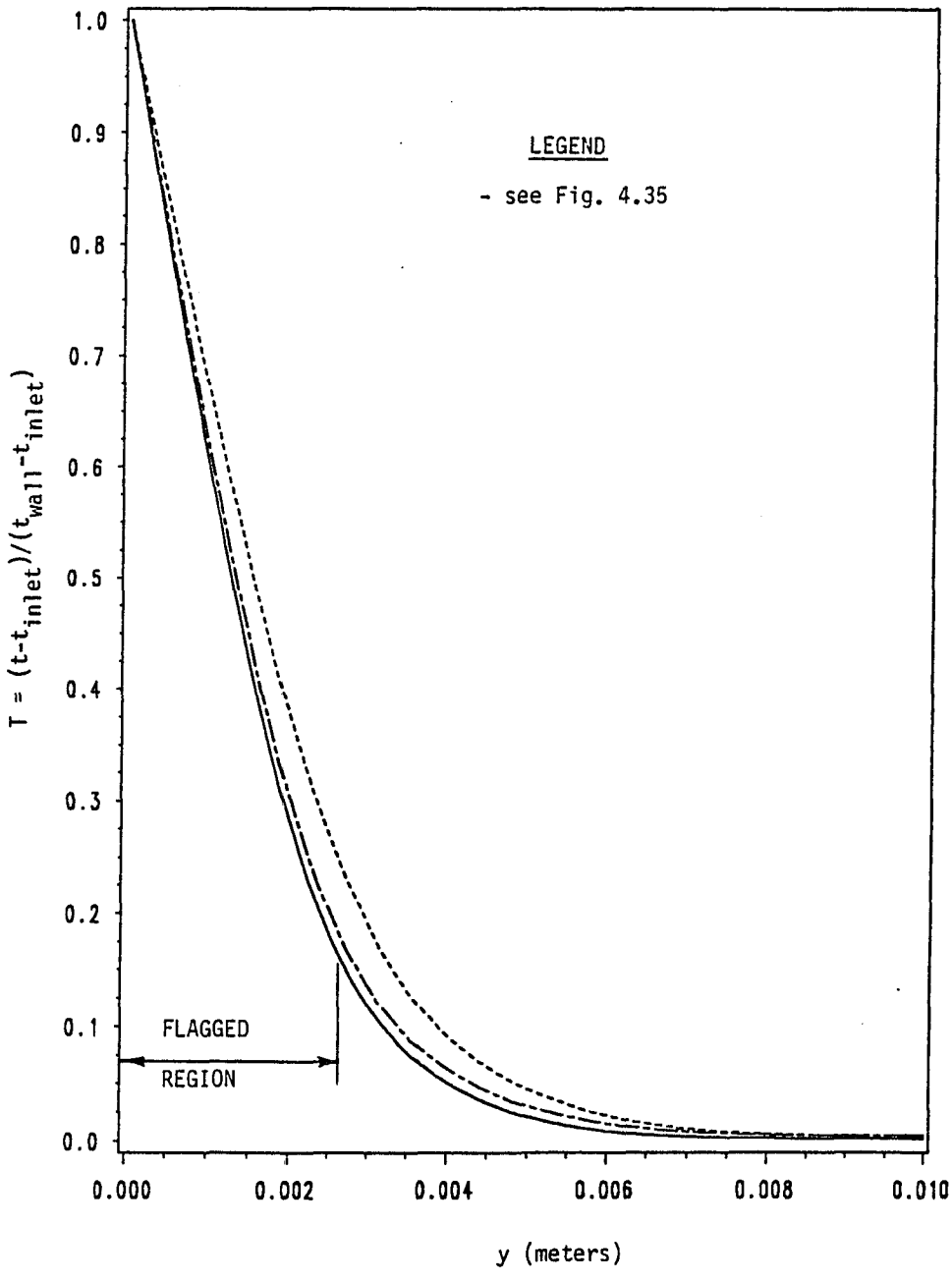


Fig. 4.37 Temperature profile at x/s = 8.52 for flow over a backward facing step problem. Comparing upwind, MAD3-FDS and fine grid QUICK solutions.

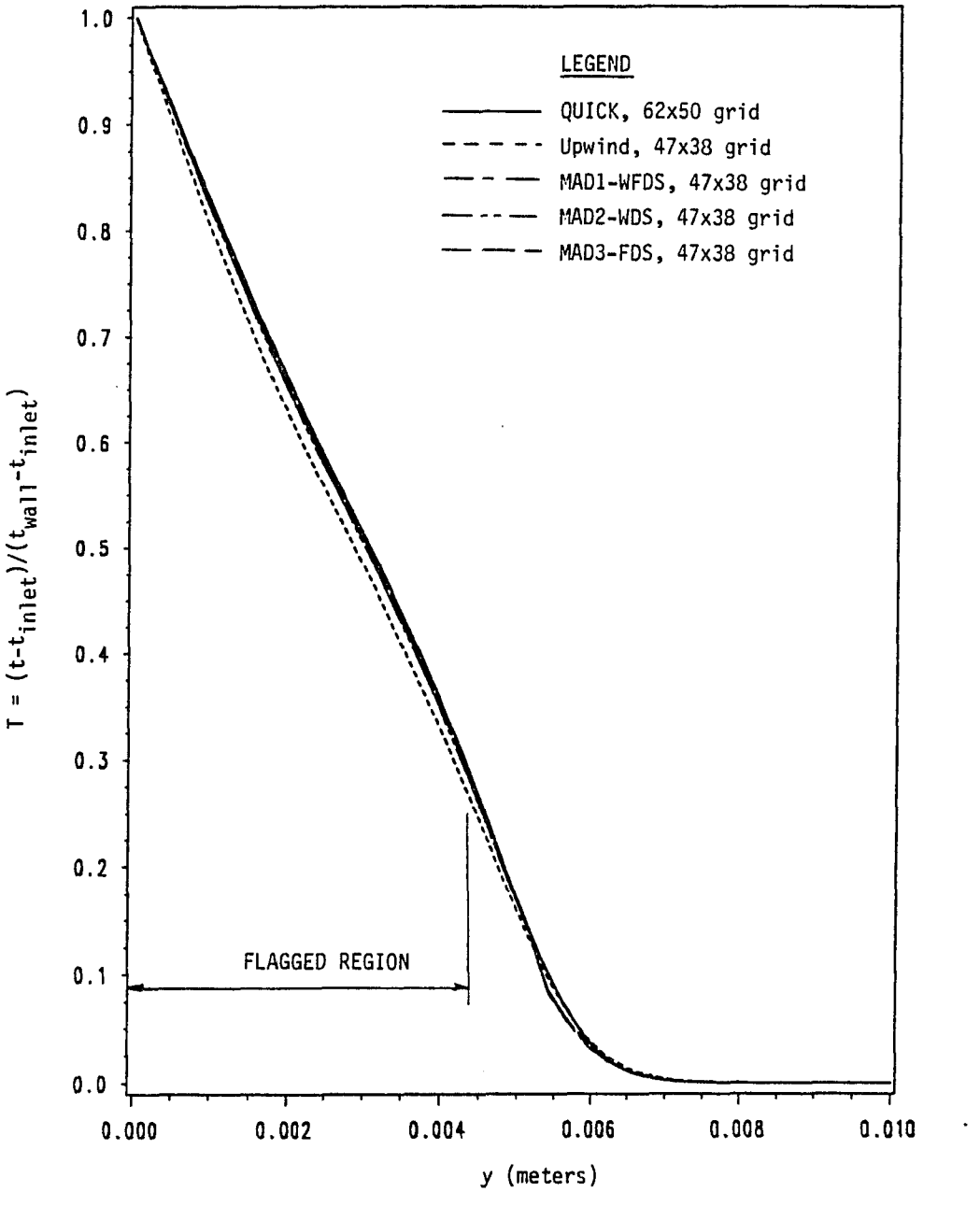


Fig. 4.38 Temperature profile at x/s = 2.55 for flow over a backward facing step problem. Comparing upwind, MAD1-WFDS, MAD2-WDS, MAD3-FDS with the fine grid QUICK solution.

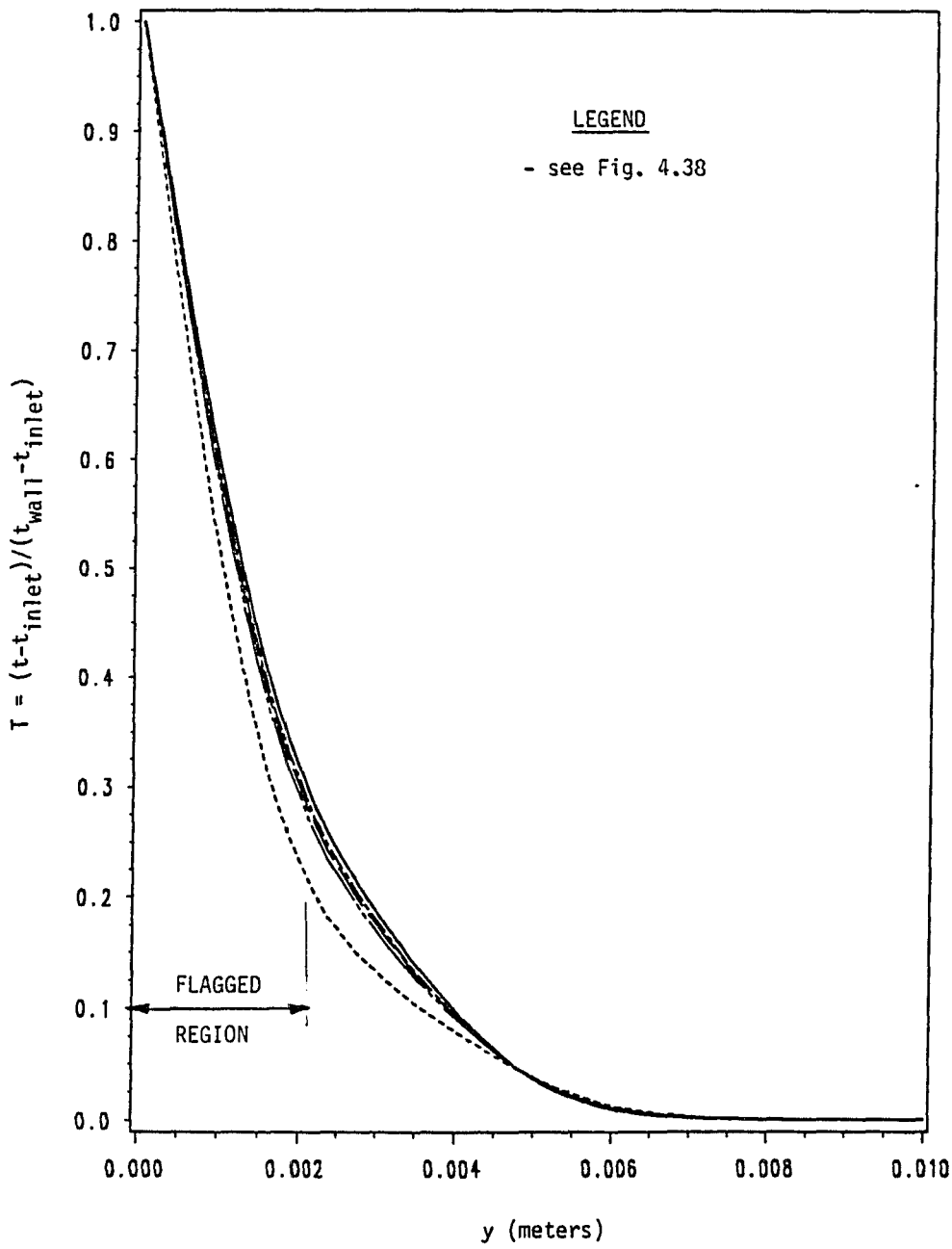


Fig. 4.39 Temperature profile at x/s = 6.12 for flow over a backward facing step problem. Comparing upwind, MAD1-WFDS, MAD2-WDS, MAD3-FDS with the fine grid QUICK solution.

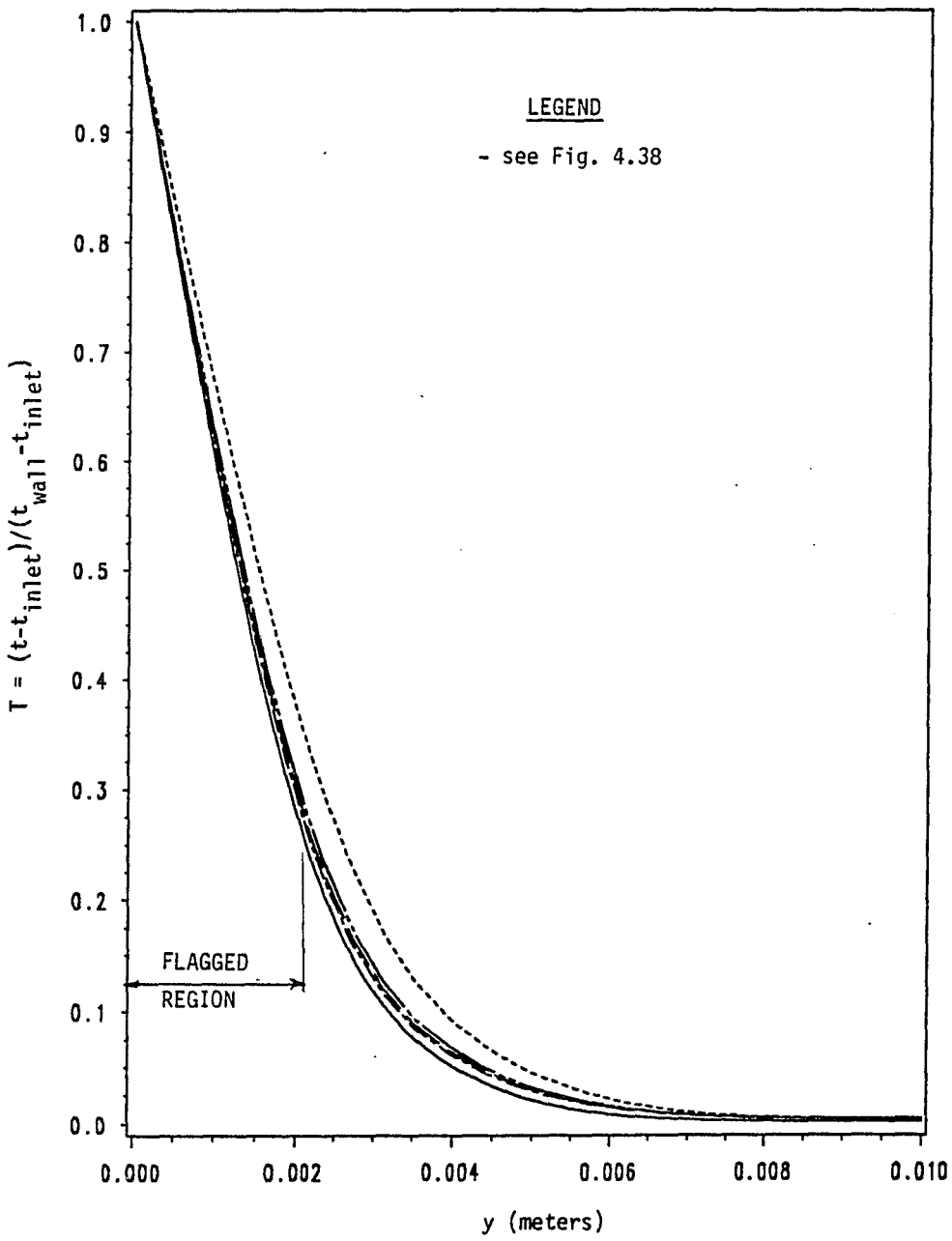


Fig. 4.40 Temperature profile at x/s = 8.52 for flow over a backward facing step problem. Comparing upwind, MAD1-WFDS, MAD2-WDS, MAD3-FDS with the fine grid QUICK solution.

Figures 4.38, 4.39 and 4.40 display the three MAD method solutions with the upwind and fine grid QUICK solutions. The three MAD temperature profiles are very similar at all three x/s locations shown. At x/s = 2.55, they very nearly overlay not only each other, but also the fine grid QUICK result.

4.7 CONCLUDING REMARKS

The solution method was extended from convection-diffusion to general flow problems. The three multiple-grid adaptive differencing schemes were applied to two standard test case problems. The results were compared and the computer timing information was presented and disucssed.

Chapter Five introduces the concept of parallelization. The MAD1-WFDS scheme is then parallelized on several levels and compared with upwind and QUICK timing results for the flow over a backward facing step problem.

CHAPTER 5

PARALLELIZATION OF AN ADAPTIVE DIFFERENCING SCHEME

5.1 INTRODUCTION

As more refined (and accurate) models are developed, the need for faster computers to solve the models in a timely manner becomes imperative. However, the limits of current technology places physical constraints on any single computer's operational speed. The current trend is to link several computers together to work in tandem. This concept is called parallel processing. According to Quinn (1987),

> "Parallel processing is a kind of information processing that emphasizes the concurrent manipulation of data elements belonging to one or more processes solving a single problem."

Vector pipelining is one method used in parallel processing. Pipelining involves streaming vectors, rather than scalars, from memory into a computer's central processing unit (CPU) where specially designed arithemetic units operate on the vectors. The other method is parallelism, which may be defined as using multiple resources to achieve concurrency. In this method, a set of independent tasks are mapped onto multiple, concurrently operating computer processors.

The majority of the work done in parallel processing has been in the finite element area. Much of the work which has been done in CFD is in the area of domain decomposition. Domain decomposition is a subject of much interest in parallel

processing because the division of a large region into smaller subdomains, which may be solved or worked upon concurrently is a natural avenue in which to exploit parallel computing capabilities. The subdomains are assigned to independent processors, but communicate information with one another through subdomain interfaces and global variables in order to maintain the solution integrity.

Hughes, et al (1987) developed a vectorized version of the element-by-element preconditioned conjugate tradients algorithm. Farhat and Crivelli (1989) present a computation strategy for nonlinear finite element computations which employs both iterative and direct solution methods. Explicit computations are carried out at the element level while implicit calculations are used at the subdomain level. They achieve speedups of 79 to 99 percent. Rai (1988) presents some of the basic ideas in domain decompositon. He discusses the transfer of information between subdomains in a multidomain problem and appplies these ideas to a simple rotor/stator interaction problem. He concludes that the domain decompositon approach facilitates 1) the use of selective grid refinement, 2) block processing, and 3) the use of different equation sets in different parts of the flow field. Wang and Gerogiadis (1990) combined curvilinear coordinate transformation with domain decommposition for modeling steady convective heat transfer in irregular geometries. They stress the importance of efficient domain

decomposition so that computer processors are not idle and waiting for other processors to complete their assigned tasks. Amon (1990) developed a vectorized and parallel implementation of the spectral element-Fourier method for the incompressible, unsteady, three-dimensional Navier-Stokes equations. This method is a domain decomposition technique which combines globally unstructured and locally structured spatial discretizations. Ecer et al (1990) developed and implemented a block-structured solution scheme for solving three-dimensional transonic flow problems on a parallel computer. Malone (1988) developed a domain decomposition algorithm which automatically divides an arbitrary finite element mesh into regions in a manner which minimizes interprocessor communication and balances computational load. He reports speed-up factors greater than 31 on a 32-processor Intel hypercube computer.

In the area of finite difference methods of computational fluid dynamics, conjugate gradient methods are receiving attention as candidates for parallel solvers. Keyes (1989) investigates domain decomposition methods for the parallel computation of reacting flows. He solves the system of partial differential equations with several finite difference methods including a psuedo-transient version of Newton iteration and preconditioned iterative methods of the conjugate gradient and Chebyshev type. He concludes that the

generalized minimum residual method with block-ILU preconditioning is the best serial method considered.

The remainder of the chapter is devoted to the discussion of the parallelization of the upwind, QUICK and MAD1-WFDS schemes. A brief description of the computer program is followed by an explanation of the levels of parallelization explored and the results achieved.

5.2 DESCRIPTION OF THE COMPUTER PROGRAM

To facilitate a disucssion of parallelizing the program for the MAD1-WFDS algorithm, a brief description of the computer program is presented here. Figure 5.1 shows the flow diagram for the MAP1-WFDS algorithm (in the serial mode). The bulk of the program is independent of the particular physical problem to be solved. The USER subroutine (indicated with a dashed line) is written by the user to describe the problem domain. A brief discussion of the functions of each subroutine follows.

Program MAIN

The function of the main program is to call the other subroutines in the proper sequence. The heart of the MAD1-WFDS logic resides here.

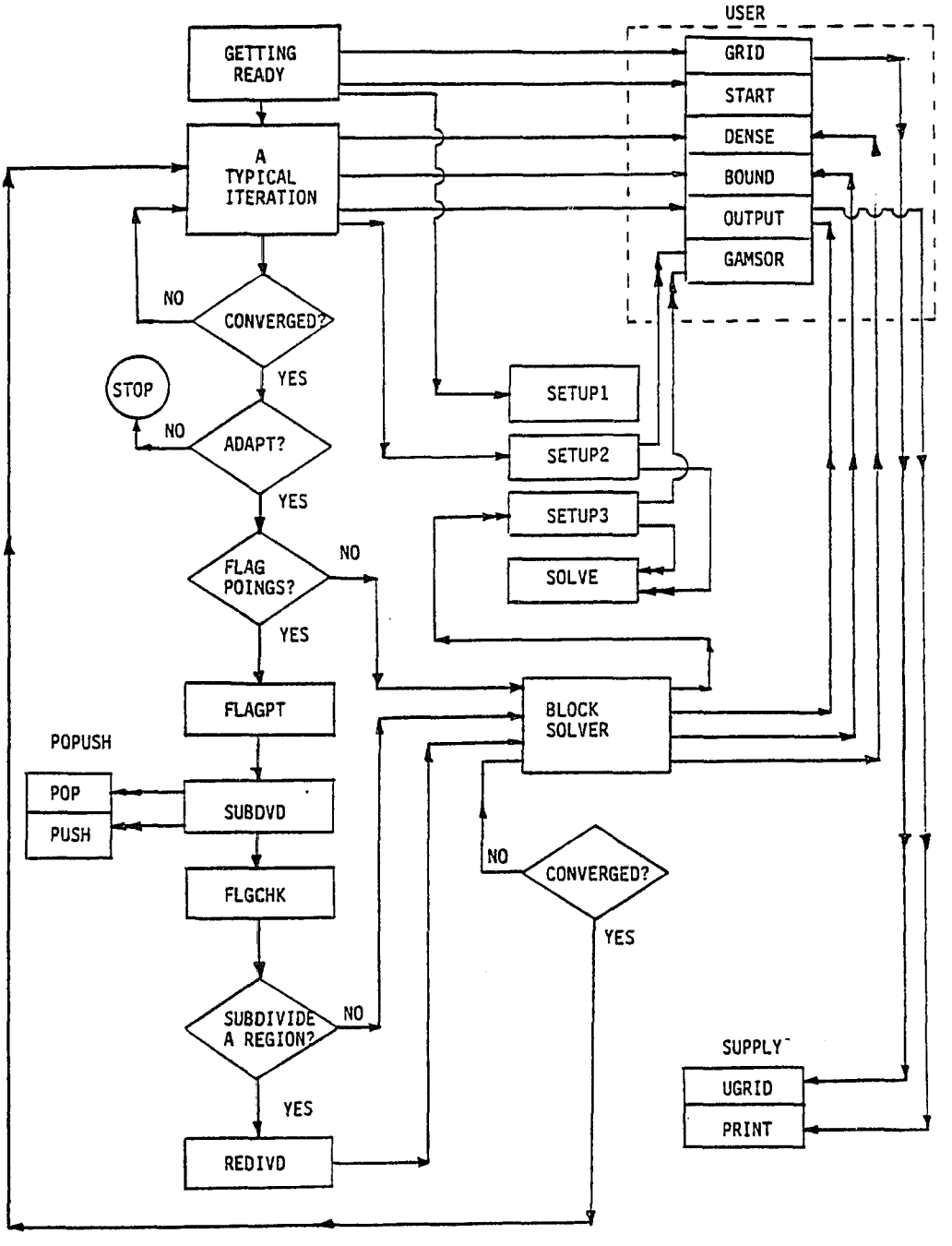


Fig. 5.1 Flow diagram for the MAD1-WFDS algorithm in the serial mode.

Subroutine SETUP1

SETUP1 calculates and stores the geometrical quantities needed for computing the equation coefficients. It is called once as a part of the start up procedure.

Subroutine SETUP2

This subroutine calculates the equation coefficients using the upwind scheme for the convection terms. The SIMPLER logic is employed in this subroutine for determining the order of equation solving. Calls to the solver subroutine are made from here.

Subroutine SETUP3

This subroutine calculates the equation coefficients using the QUICK scheme for the convection terms. The SIMPLER logic is employed again in this subroutine for determining the order of equation solving. Calls to the solver subroutine are also made from here.

Subroutine SOLVE

The discritized equations are solved in this subroutine using a line-by-line Tri-Diagonal Matrix Algorithm. Prior to employing the TDMA, a block correction procedure is used to accelerate the solution convergence.

Subroutine SUPPLY

This subroutine has two entry points, UGRID and PRINT. Entry UGRID generates a uniform grid given the domain limits and the desired number of grid points along each axis. Entry PRINT prints out the grid point locations and the solved

variable arrays. Boundary pressure values are calculated by extrapolation before printing the pressure array.

Subroutine FLAGPT

The normalized error array is calculated according to Eq. 3.3. and then grid points flagged accordingly.

Subroutine SUBDVD

This subroutine locates clusters of flagged grid points and defines the number of flagged regions.

Subroutine FLGCHK

Each flagged region is examined for size. If the region spans more than a user-supplied maximum percent of the problem domain, it is divided in half (see Subroutine REDIVD). If any region is larger by a user defined factor than another region, the larger region is divided in half.

Subroutine REDIVD

This subroutine divides a flagged region in half and adjusts the necessary bookkeeping quantities.

Subroutine POPUSH

This subroutine is called by FLAGPT to aid in identifying clusters of flagged grid points. It is composed of the standard pop and push algorithms used on stack data structures.

Subroutine USER

This subroutine defines the problem to be solved and is comprised of several entry points. Entry GRID supplies the program with the problem geometry and grid. Entry START gives

the initial values for all variables. Both GRID and START are called once during the startup procedure. The remaining four entry points, DENSE, BOUND, OUTPUT and GAMSOR are called at each iteration. The density is supplied in Entry DENSE while boundary values are updated (if necessary) in Entry BOUND . Any values of interest may be printed out in Entry OUTPUT at each iteration. Finally, in Entry GAMSOR the diffusion coefficient, Γ, and the source term coefficients, S_c and S_p, are specified for each dependent variable.

5.3 PARALLELIZATION OF THE CODE AND RESULTS

The problem of flow over a backward facing step presented in Chapter Four is modified for use as a parallel test case. MAD1-WFDS was chosen as the solution scheme since it not only yielded the most improved results, but it also lends itself to more levels of parallelization. The continuity and x- and y-momentum equations remain unchanged. However, four different sets of temperature field boundary conditions are now to be solved simultaneously. These four temperature fields are solved after a converged velocity field is obatined. The first temperature field case is the same as that in Chapter Four. The other three sets of boundary conditions are listed below:

Case 2

$T = 0.0$ for $y > s$ at the inlet

$T = 1.0$ for $y < s$ at the inlet

$\partial T/\partial y = 0.0$	at the outflow boundary
$T = 1.0$	along the bottom wall
$q'' = 0.0$	along the top wall

Case 3

$T = 0.0$	at the inlet
$\partial T/\partial x = 0.0$	at the outflow boundary
$q'' = -k\ \partial T/\partial y$	along the bottom wall
$q'' = 0.0$	along the top wall

Case 4

$T = 0.0$	at the inlet
$\partial T/\partial x = 0.0$	at the outflow boundary
$q'' = 0.0$	along the top wall
$q'' = h(T - T_{inf})$	along the bottom wall

The problem solution was obtained on a 102 x 102 grid for each of the computer runs made.

The computer used for this research is the IBM 3090-600J. This 3090 has six central processing units, each with its own attached vector processor. Thus up to six flagged regions may be concurrently solved (each on its own processor) simultaneously in parallel on an unloaded (or uncommitted) system.

In a multiuser system many users' jobs are excuted concurrently. This is accomplished by placing the users' jobs in a queu. When an individual user's job reaches the top of a queu, it recieves time on an available processor. Usually, a job will occupy a processor for a short time and then be

"rolled" back out and placed in the queu to await the next opportunity for mor processor time. This occurs repeatedly until the job is complete.

In discussing the timing information, the term virtual cpu time refers to the actual amount of time a job accrues on the computer's processor(s). Total cpu time, however, may be thought of as the virtual computing time plus overhead associated with rolling a job in and out of the processor and queu. Real or wall time is actual clock time from the start of a computer job until its completion. This time includes all cpu time and idle time spent waiting in queus.

Several different runs were made to explore the levels of parallelization. Initially, however, the problem was vectorized and run in a serial mode using the upwind, QUICK and MAD1-WFDS schemes as base or control run times. The upwind scheme required 6,902.41 seconds of total cpu time (this total cpu time includes the time to solve the flow and temperature fields), while the QUICK method consumed 12,320.81 seconds. The MAD1-WFDS algorithm used 8,742.18 seconds.

There are several levels of the program which are natural canditates for parallelization. First, any flagged subregions may be solved in parallel. Second, within the subroutines SETUP2 and SETUP3, the general ϕ equations may be solved simultaneously. Finally, and also within SETUP2 and SETUP3, the u and v velocity equation coefficients may be caculated concurrently and solved at the same time. Before any of these

options were explored, however, a parallel version of the equation solver was installed. The serial version of the equation solver (line-by-line TDMA) used a Gauss-Seidel approach. To parallelize this portion of the code, the Gauss-Jordan scheme was used instead. In this manner, blocks of the problem domain can be solved concurrently.

There may be one or more flagged regions resulting from the error estimate analysis. The local higher order (QUICK) solution in these regions are independent of one another since each subdomain has its own set of boundary conditions (provided by the current upwind solution and /or global domain boundary values). These self-contained, independent regions may be solved simultaneously. To take full advantage of the IBM 3090's parallel capability, the flagged regions are examined for comparable size. If any region is equal to or larger than the other by some user input factor (usually 2), then the larger region is further subdivided, balancing the computer's workload and minimizing idle processor time. Or if a flagged region exceeds a user input maximum size, it is divided in half. For this problem, if any one side of a flagged region exceeded fifty percent of the global domain length along either the x- or y-axis, it was subdivided. If a large flagged region has been divided into two subregions, then a boundary is created within the flagged region (see Fig. 5.2a). Since an improved QUICK solution is desired at all the flagged grid points, such an interior boundary is undesirable.

This problem is overcome by overlapping the two subregions in the manner shown in Fig. 5.2b. Now the west boundary for Subregion 1 lies within Subregion 2 and the east boundary of Subregion 2 is within Subregion 1. In this manner, an improved solution is obtained at each flagged grid point. The values used along these adjoining interior boundaries may be handled in one of two ways. First, the most recent upwind value obtained prior to adaptation may be used as a constant boundary condition . Second, the most current value supplied by the adjoining region may be used as a dynamic boundary value. Since an initial converged upwind value is supplied so that solution divergence is not a problem, the second approach is adopted.

The timing information for this parallization case is presented in Table 5.1. Since the upwind and QUICK schemes do not have any flagged regions, the information is for the MAD1-WFDS scheme only. The total cpu time for the entire job (including calculating the flow field) is 9,201.33 seconds. This time may be compared with the serial time of 6,902.41 seconds for the upwind scheme and 13,320.81 seconds for the QUICK solution (see Table 5.4). The parallel virtual processors (namely, vcpu0, vcpu1, vcpu2 and vcpu3) were initiated at the start of the adaption process (after obtaining the initial upwind velocity field). The time associated with vcpu 0 includes the time required to obtain the converged upwind temperature field after the converged

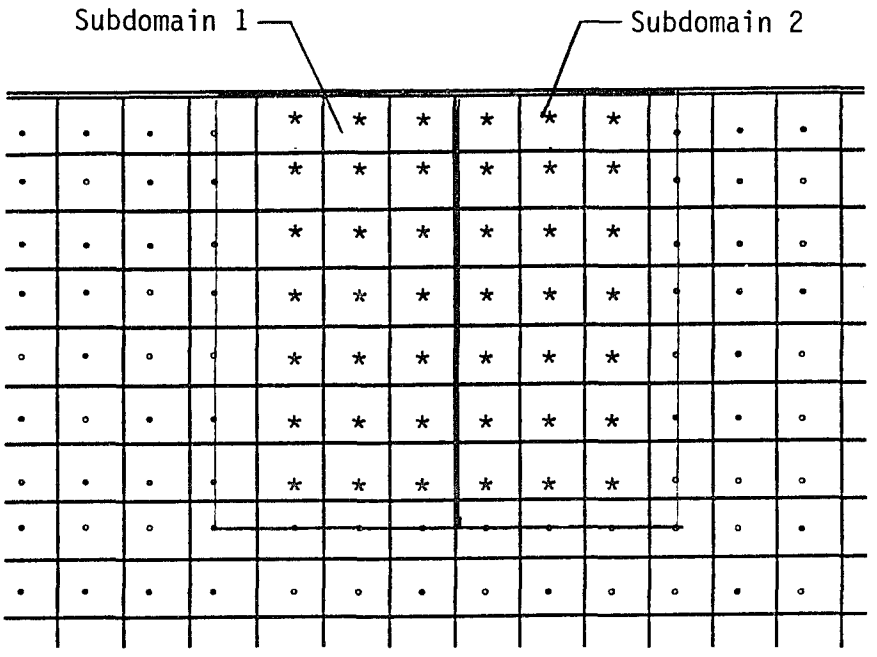


Fig. 5.2a Division of a large flagged subdomain into subregions.

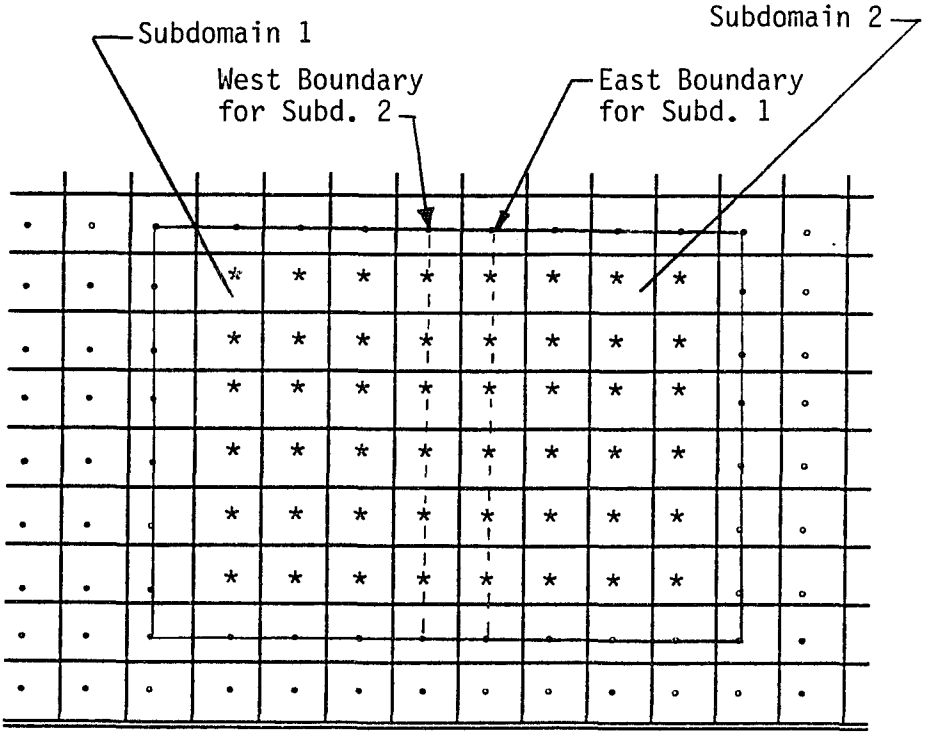


Fig. 5.2b Overlapping subregions in a flagged subdomain.

Processor	Virtual cpu (sec)	Total cpu (sec)
MAD1-WFDS: Total for entire job	8,798.52	9,201.33
vcpu 0	4,448.0	4,651.0
vcpu 1	662.0	667.0
vcpu 2	668.0	675.0
vcpu 3	546.0	550.0

Note: The virtual processors were initialized after the temperature field was flagged at the start of the adaption process.

Table 5.1 Computer timing information for parallelizing at the flagged subdomain level.

velocity field solution was computed. It also includes the time spent in sweeping the global domain after each set of subdomain solutions for both the velocity and temperature fields. The velocity field flagging criteria produced two flagged regions (see Fig. 4.21). The larger flagged region along the south boundary was subdivided into two subregions, making a total of three flagged regions. The time required to solve the velocity and pressure fields in the three flagged regions using the QUICK method is included the time accrued by vcpu1, vcpu2 and vcpu3. The temperature field flagging criteria, however, resulted in one large flagged region (see Fig. 4.34), which was subdivided into two flagged subregions. Hence, virtual processors (vcpu) 1 and 2 have more time than the third processor. However, the load does appear to be fairly well distributed, indicating a sasisfactory domain decomposition scheme.

Before presenting the next parallelization case, a closer scrutiny of subroutines SETUP2 and SETUP3 is provided. As stated previously, the SIMPLER logic provides the basic structure for these two subroutines. Fig. 5.3 shows a flow chart for SETUP2 (which is equally applicable to SETUP3). In the original serial version of the program, the coefficients for the u velocity are calculated and stored first. Next, the v-velocity equation coefficients are computed and stored, and then the coefficients for the pressure equation determined.

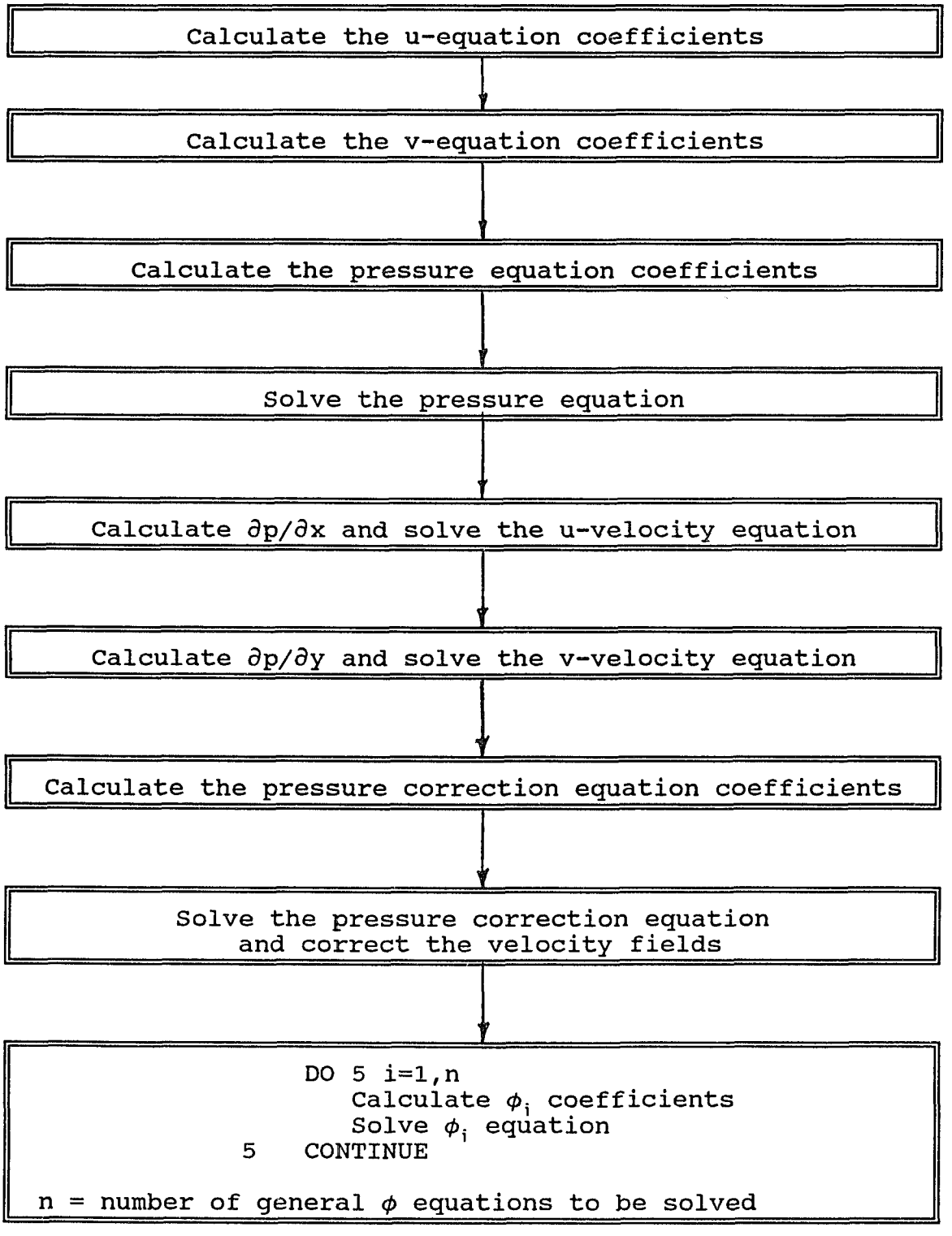


Fig. 5.3 Flow chart for SETUP2 in the serial mode.

The coefficients for the two velocity equations are calculated prior to the pressure equation coefficients since the u and v quantities (see Eq. 4.16) are required in the source term of the pressure equation. The pressure equation field is now solved. With the pressure field newly updated, the values of $\partial p/\partial x$ and $\partial p/\partial y$ may be computed and the two velocity equation fields solved. At this point, the newly updated velocity values are used in computing the pressure correction equation coeficients, and the pressure correction equation is solved. Now the velocity fields may be corrected via Eq. 4.10. Finally, any remaining general ϕ variable fields are determined by calculating the coefficients and calling the solver within a loop. This last step is the candidate for the next level of parallelization.

Since the four temperature field equations depend upon the velocity fields, but not upon each other, they may be solved independently and simultaneously. Table 5.2 shows the resulting computer times for this parallelization case for the upwind, QUICK and MAD1-WFDS schemes. In the upwind scheme, the real (or wall) time consumed while the four temperature fields were computed was 2,423.60 seconds, while the total virtual cpu time consumed by the four processors (each computing one of the temperature fields) was 3,295.0 seconds. This is a savings of 871.4 real seconds or 26.4%. Note that this savings in terms of real or wall time occurred even though the job was <u>not</u> run on a dedicated machine. Comparing

Processor	Virtual cpu (sec)	Total cpu (sec)	# of Cyc.	# of Iter	cpu/ Iter
Upwind: Total for job	7,039.50	7,440.49	---	2500	2.98
vcpu 0	2.0	2.0	---	---	---
vcpu 1	824.0	831.0	---	1000	0.83
vcpu 2	825.0	833.0	---	1000	0.83
vcpu 3	821.0	828.0	---	1000	0.83
vcpu 4	823.0	830.0	---	1000	0.83
QUICK: Total for job	13,327.52	14,310.23	---	3500	4.09
vcpu 0	3.0	3.0	---	---	---
vcpu 1	951.0	960.0	---	1000	0.96
vcpu 2	949.0	957.0	---	1000	0.96
vcpu 3	952.0	961.0	---	1000	0.96
vcpu 4	954.0	963.0	---	1000	0.96
MAD1-WFDS: Total for job	9,016.47	9,368.52	5/3	3000	3.12
vcpu 0	983.0	991.0	3	1000	0.91
vcpu 1a	80.0	82.0	3	150	0.55
vcpu 1b	81.0	83.0	3	150	0.55
vcpu 1c	79.0	80.0	3	150	0.54
vcpu 1d	80.0	82.0	3	150	0.55
Total Subd. 1	320.0	327.0	---	600	0.55
vcpu 2a	81.0	83.0	3	150	0.55
vcpu 2b	82.0	84.0	3	150	0.56
vcpu 2c	80.0	83.0	3	150	0.55
vcpu 2d	81.0	83.0	3	150	0.55
Total Subd. 2	324.0	333.0	---	600	0.55

Table 5.2 Computer timing information for the parallelization of the temperature equations.

the total cpu times for the serial and parallel temperature equation cases (using the upwind scheme) only a 7.8% increase in the total cpu time was required to run the job in parallel (due to overhead associated with initiating and scheduling tasks in parallel) while a 26.4% real time savings was obtained. The computing load is evenly distributed among the four processors for all three solution schemes, as indeed it should be, since each processor is performing essentially the same task - solving a temperature field. For the MAD1-WFDS scheme, an upwind temperature field is initially computed and flagged. The flagged region is divided into two subregions as before. Then, each subregion solves the four temperature problems in parallel for a total of eight equations being solved in parallel at each multiple-grid iteration. The individual processor times (for vcpu 0, vcpu 1a - vcpu 1d, and vcpu 2a - vcpu 2d) were initialized at the time the temperature fields were flagged and the adaption process began. The time for vcpu 0 (991 total cpu seconds) includes the time required for the global domain solution between subdomain solutions.

Finally, the procedure for the calculation of the velocity and pressure fields was examined for parallelization possibilities. In addition to parallelizing the flagged region solutions and the temperature field solutions, the velocity and pressure field solution scheme (SIMPLER) is to be parallelized. The coefficients for the u- and v-velocity

equations are independent of one another and so are setup to be computed concurrently. These equations are also solved simultaneously. Figure 5.4 shows the new parallel flow chart for SETUP2 (or SETUP3). The flow chart would be applicable to the upwind, QUICK or any of the MAD schemes. The timing results are presented in Table 5.3 for the upwind scheme. Note the dramatic increase in overall total cpu time required for this run as compared with the parallel temperature case for the upwind scheme - an increase in time of 61%. Clearly this an unaccpetable increase in cost to achieve further parallelization. This increase in cpu time required is due to the increased overhead costs and idle processor time. This parallelization case was not run for the QUICK or MAD1-WFDS schemes because of the undesirable results obtained with the upwind scheme.

Table 5.4 compares the total cpu time required by each parallel job for a given solution scheme with its own base run time according to the following formula:

$$\%\ Increase\ in\ cpu\ time = \frac{Time_{parallel\ case} - Time_{serial\ case}}{Time_{serialcase}} \times 100$$

The recommended level of parallelization to achieve maximum real time savings (especially if run on a dedicated system) with minimal increase in overall cpu time is to parallelize the four temperature equations for the upwind, QUICK and MAD1-WFDS schemes and to parallelize the flagged subdomain computations for the MAD1-WFDS scheme.

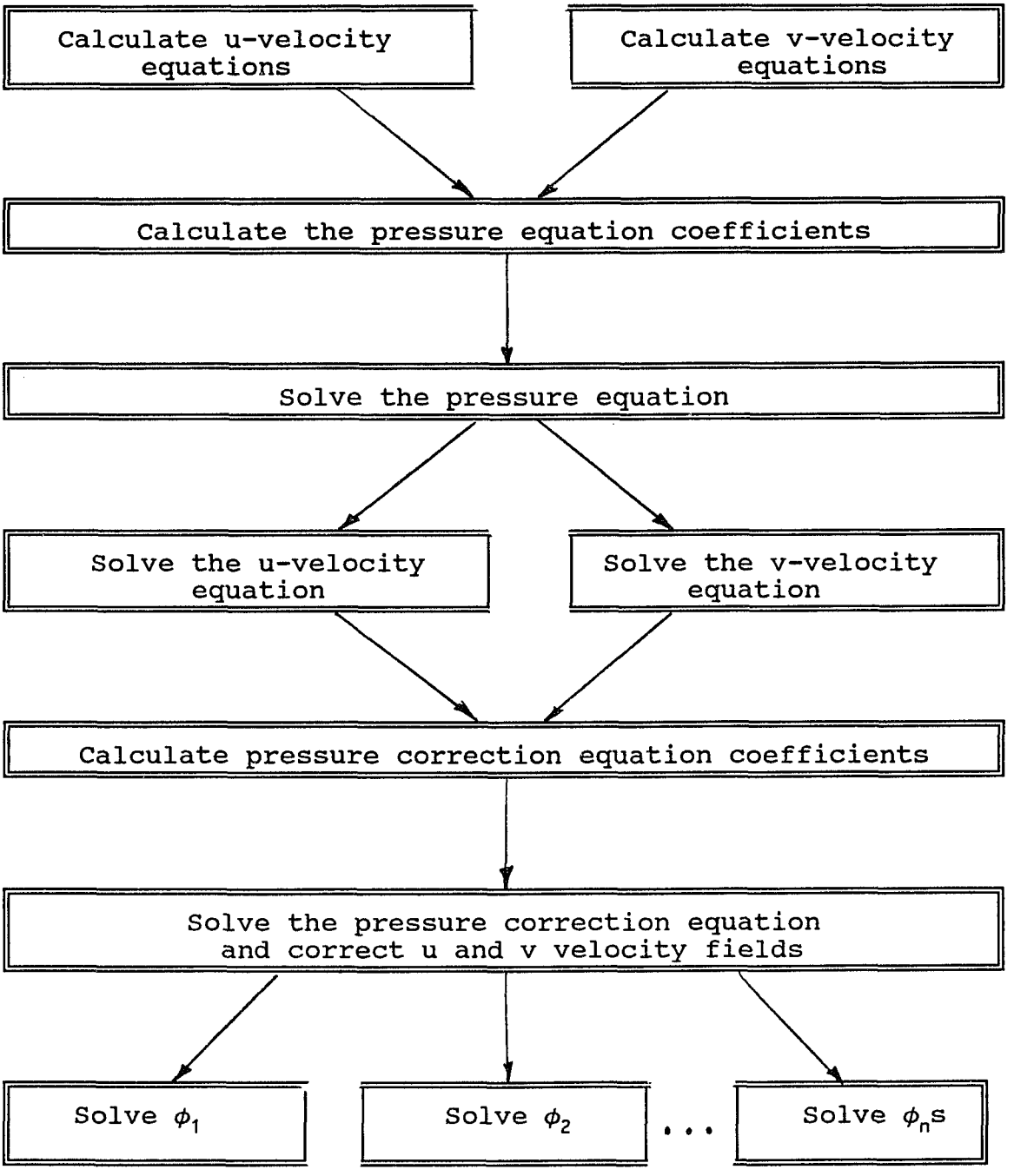


Fig. 5.4 Flow chart for parallel version of subroutine SETUP2

Processor	Virtual cpu (sec)	Total cpu (sec)
Upwind: Total of all processors	11,343.41	12,735.33
vcpu 0	1,688.0	2,435.0
vcpu 1	2,316.0	2,474.0
vcpu 2	2,114.0	2,314.0
vcpu 3	2,171.0	2,314.0
vcpu 4	3,053.0	3,261.0

Table 5.3 Computer timing information for parallelizing the velocity and pressure equations.

Scheme	Parallelization Case	Total cpu (sec)	% increase in cpu
Upwind	Serial	6,902.41	-
	Parallel temp. fields	7,440.49	7.8
	Parallel vel. & temp. fields	12,735.33	84.5
QUICK	Serial	13,320.81	-
	Parallel temp. fields	14,310.23	7.4
MAD1-WFDS	Serial	8,742.18	-
	Parallel Subregions	9,201.33	5.3
	Parallel subdomains & temp. fields	9,368.52	7.2

Table 5.4 Comparison of parallel and serial total cpu run times for upwind, QUICK and MAD1-WFDS solution schemes.

5.4 CLOSING REMARKS

In this chapter, the concept of parallization was introduced and the upwind, QUICK and MAD1-WFDS schemes were parallelized at several levels. Test case results were obtained using a modified version of the flow over a backward facing step problem. Parallelizing the solution of the four temperature equations yielded desirable results. Parallelizing the solution of the flagged subdomains using the domain decomposition technique described herein also produced desirable results.

The next chapter provides a brief summary of the work incorporated within this dissertation and suggestions for future work.

CHAPTER 6

CONCLUDING REMARKS AND FUTURE TASKS

6.1 REVIEW OF THE WORK

Two objectives were presented. The first is to develop an adaptively-differencing multiple-grid finite difference algorithm for the solution of the incompressible Navier-Stokes equation. The second objective is to perform a study of parallelizing the algorithm to reduce the real computation time required for problem solutions.

A general purpose finite difference method for fluid flow and heat transfer problems is developed. The method is presented first for convection-diffusion problems. The discretization of the convection terms in both the general ϕ equation and the momentum equations is presented using two methods, upwind and QUICK. The method is extended to general fluid flow problems, where the flow field is not known a-priori. The SIMPLER algorithm is introduced for the solution of the pressure and velocity fields.

Three adaptive differencing schemes are developed and applied to two convection-diffusion problems and two fluid flow problems. A scheme for flagging regions of high error estimate is introduced and applied to an initial upwind solution over the problem domain. At this point, any one of the three adapting schemes, MAD1-WFDS, Mad2-WDS or MAD3-FDS, may be applied. The solutions are examined for improvements

over the upwind solution and the computer effort required to achieve those enhancements. The MAD1-WFDS scheme provided the best improvement to the upwind solution and also required the most computing effort. MAD2-WDS required the least increase in cpu time (compared to the upwind scheme) and also produded the least improvements to the solution.

The MAD1-WFDS algorithm was selected for parallelization. The upwind and QUICK algorithms were parallelized for comparison purposes. The codes were first vectorized and a parallel version of the discretized equation solver implemented. Several program levels or granularities were parallelized. In the case of parallelizing the four temperature equations for the upwind scheme, a savings of 26.4% (871.4 seconds) in real (or wall) time was achieved in spite of the fact that the parallel job was not run on a dedicated machine. The parallel version of the upwind code required only a 7.8% increase in the total cpu time over the serial version. Parallelizing the solution of the velocity and pressure fields, however, resulted in an increase in cpu time of 61% for the upwind scheme. This level of parallelization is not recommended. In general, parallelizing the QUICK solution of the flagged subdomains resulted in efficient use of the parallel processors. Halving "large" flagged regions proved to be an additional effective domain decomposition technique. Likewise, parallelizing the solution

of general ϕ equation (as in the four temperature equations) also makes effective use of the parallel processors.

6.2 FUTURE WORK

Much of the computing effort is spent in solving the systems of discretized equations. Many parallel partial differential equation solvers have been developed and implemented with varying degrees of success. One promising solver used is the preconditioned conjugate gradient method. Ventakrishnan (1990) uses this algorithm with great success to solve a variety of two-dimensional viscous and inviscid Navier-Stokes problems. This equation solving scheme and others should be investigated for possible (probable) reduction in cpu times.

Recall from the Introduction that in finite element literature, adaptive grid methods which refine the mesh are classified as h-methods while adaptation schemes which increase the leocal degree of the polynomial shape function are referred to as p-methods. There has been considerable interest of late in algorithms which combine h- and p-methods, referred to as h-p methods. Devloo et al (1988) present a combined h-p finite element adaptive strategy for accurately modeling the Navier-Stokes equations of viscous compressible fluid flow in two dimensions. Oden et al (1991) developed a finite element strategy with the mesh size and discretization order as control parameters. They obtained exponential rates

of convergence when applying this method to some representative problems in compressible viscous flow. Combining an h-type method with the p-type method developed in this dissertation should be investigated for possible acceleration of the solution convergence.

The coeficient α_3 in Eq. 3.2 was set to zero for the test problems in this dissertation. A study of the rate of change of the gradient of the dependent variable of the flagging scheme should be made.

The adaptation schemes developed herein may be readily extended to three dimensions. Such an enhancement to the program will make it more applicable to a wider variety of problems.

Parallelizaton of the MAD2-WDS and MAD3-FDS will make these programs more efficient and less real time consuming for large problems.

BIBLIOGRAPHY

Acharya, S. and Patankar, S.V. (1982), "Use of An Adaptive Grid for Parabolic Flows," AIAA Paper 82-1015 presented at the AIAA/ASME 3rd Joint Thermophysics, Fluids, Plasma and Heat Transfer Conference, St. Louis, Missouri.

Acharya, S. and Moukalled, F.H. (1990), "An Adaptive Grid Solution Procedure for Convection-Difussion Problems," Journal of Computational Physics, vol. 91, pp. 32-54.

Amon, C.H. (1990), "Spectral Element-Fourier Approximation for the Navier-Stokes Equations: Some Aspects of Parallel Implementation," presented at AIAA/ASME Thermophysics and Heat Transfer Conference, Seattle, Washington.

Anderson, D.A. and Rai, M.M. (1982), "The Use of Solution Adaptive Grids in Solving Partial Differential Equations," Numerical Grid Generation, ed. J.F. Thompson, North-Holland.

Armaly, B.F., Durst, F., Pereira, J.C.F. and Schonung, B., (1983), "Experimental and Theoretical Investigation of Backward-Facing Step Flow," Journal of Fluid Mechanics, vol. 127, pp. 473-496.

Berger, M.J. and Jameson, A. (1985), "Automatic Adaptive Grid Refinement for the Euler Equations," AIAA Journal, vol. 23, pp. 561-568.

Braaten, M.E. and Shyy, W. (1987), "Study of Pressure Correction Methods with Multigrid for Viscous Flow Calculations in Nonorthogonal Curvilinear Coordinates," Numerical Heat Transfer , vol. 11, pp. 417-442.

Brackbill, J.U. (1982), "Coordinate System Control: Adaptive Meshes," Numerical Grid Generation, ed. J.F. Thompson, North-Holland.

Brandt, A. (1977), "Multi-Level Adaptive Solutions to Boundary Value Problems," Math. Comput., vol. 31, pp. 333-390.

Burggraf, O.R. (1966), "Analytical and Numerical Studies of the Structure of Steady Separated Flows," Journal of Fluid Mechanics, vol.24, pp. 113-151.

Caruso, S.C., Ferziger, J.H., and Oliger, J. (1986), "Adaptive Grid Techniques for Elliptic Fluid Flow Problems," AIAA 24th Aerospace Meeting, Reno, Nevada.

Demkowicz, L., Oden, J.T., Strouboulis, T. and Devloo, P. (1985), "An Adaptive p-Version Finite Element Method for Transient Flow Problems with Moving Boundaries," Finite Elements in Fluids, vol. 6, eds. R.H. Gallagher, G.F. Carey, J.T. Oden and O.C. Zienkiewicz; John Wiley and Sons, Ltd., Chichester, pp. 291-305.

Dennis, S.C.R. and Hudson, J.D. (1989), "Compact h^4 Finite Difference Approximations to Operators of Navier-Stokes Type," Journal of Computational Physics, vol. 85, pp. 390-416.

Devloo, P., Oden, J.T., and Pattani, P. (1988), "An h-p Adaptive Finite Element Method for the Numerical Simulation of Compressible Flow," Computer Methods in Applied Mechanics and Engineering, vol. 70, pp. 203-235.

de Vahl Davis, G. and Mallinson, G.D. (1972), "False Diffusion in Numerical Fluid Mechanics," University of New South Wales, School of Mechanical and Industrial Engineering Report 1972/FMT/1.

De Zeeuw, P.M. (1990), "Matrix-Dependent Prolongations and Restrictions in a Blackbox Multigrid Solver," Journal of Computational and Applied Mathematics, vol. 33, pp. 1-27.

Dwyer, H.A., Smooke, M.D. and Kee, R.J. (1982), "Adaptive Gridding for Finite Difference Solutions to Heat and Mass Transfer Problems," Numerical Grid Generation, ed. J.F. Thompson, North-Holland.

Ecer, A., Spyropoulos, J.T. and Chang, S.M. (1990), "Parallel Computation of Three-Dimensional Transonoc Flow Problems with Complex Geometries," AIAA paper # 90-0336 presented at the 28th Aerospace Sciences Meeting, January, 1990 in Reno, Nevada.

Eiseman, P.R. (1983), "Adaptive Grid Generation by Mean Value Relaxation," Advances in Grid Generation, ASME Fluids Engineering Conference, Houston, Texas.

Eiseman, P. (1987), "Adaptive Grid Generation," Computer Methods in Applied Mechanics and Engineering, vol. 64, pp. 321-376.

Farhat, C. and Crivelli, L. (1989), "A General Approach to Nonlinear Finite Element Computations on Shared-Memory Multiprocessors," Computer Methods in Applied Mechanics and Engineering, vol. 72, pp. 153-171.

Fuchs, L. and Zhao, S. (1984), "Local Mesh-Refinement Technmique for Inocmpressible Flows,' International Journal of Numerical Methods in Fluids, vol. 4.

Ghia, U. Ghia, N. and Shin, C.T. (1982), "High-Re Solutions for Incompressible Flow Using Navier-Stokes Equations and a Multi-Grid Method," Journal of Computational Physics, vol. 48, pp. 387-411.

Gnoffo, P.A. (1982a), "A Vectorized, Finite-Volume, Adaptive Grid Algorithm Applied to Planetary Entry Problems," AIAA Paper 82-1018 presented at AIAA/ASME 3rd Joint Thermophysics, Fluids, Plasma, and Heat Transfer Conference, St. Louis, Missouri.

Gnoffo, P.A. (1982b), "A Vectorized, Finite-Volume, Adaptive-Grid Algorithm for Navier-Stokes Calculations," Numerical Grid Generation, ed. J.F. Thompson, North-Holland.

Greenburg, J.B. (1983), "A New Self-Adaptive Grid Method," AIAA Paper 83-1934 presented at AIAA 6th Computational Fluid Dynamics Conference, Danvers, Massachusettes.

Guo and Babuska, I. (1986), "The h-p Version of the Finite Element Method, Part 1," Computational Mechanics, vol. 1, pp. 21-42.

Hackbusch, W. and Trottenberg, U. (eds.) (1982), "Lecture Notes in Mathematics," Multigrid Methods, vol. 960, Springer-Verlag, Berlin.

Han, T., Humphrey, J.A.C. and Launder, B.E. (1981), "A Comparison of Hybrid and Quadratic-Upstream Differencing in High Reynolds Number Elliptic Flows," Computer Methods in Applied Mechanics and Engineering, vol. 29, pp. 81-95.

Harvey, A.D., Acharya, S., and Lawrence, S.L., (1991a), "A Solution Adaptive Grid Procedure for the Three Dimensional Parabolized Navier Stokes Solver," AIAA Paper 91-0104, presented at Reno, NV.

Harvey, A.D., Acharya, S., and Lawrence, S.L., (1991b), "Prediction of Complex Three Dimensional Flowfields by a Solution Adaptive Mesh Algorithm," AIAA Conference, Baltimore, MD.

Hirsh, R.S. (1983), "Higher Order Approximations in Fluid Mechanics - Compact to Spectral," Von Karman Institute for Fluid Dynamics, Burssels, Belgium, Lecture Series 1983-04. Compuational Fluid Dynamics, March 7-11, 1983.

Hsu, F.H. (1981), "A Curvilinear-Coordinate Method for Momentum, Heat and Mass Transfer in Domains of Irregular Geometry," Ph.D. Thesis, University of Minnesota.

Hutchinson, B.R. and Raithby, G.D. (1986), "A Multigrid Method Based on the Additive Correction Strategy," Numerical Heat Transfer, vol. 9, pp. 511-538.

Hutchinson, B.R., Galpin, P.F. and Raithby, G.D. (1988), "Application of Additive Correction Multigrid to the Coupled Fluid Flow Equations," Numerical Heat Transfer, vol. 13, no. 2, pp 133-147.

Jordan, H.F. (1987), "Interpreting Parallel Processor Performance Measurements," SIAM Journal of Sci. Stat. Comput., vol. 8, no. 2.

Kallinderis, Y.G. and Baron, J.R. (1989), "Adaption Methods for a New Navier-Stokes Algorithm," AIAA Journal, vol. 27, no. 1.

Keyes, D.E. (1989), "Domain Decomposition Methods for the Parallel Computation of Reacting Flows," Computer Physics Communications, vol. 53, pp. 181-200/

Kim, H.J. and Thompson, J.F. (1990), "Three-Dimensional Adaptive Grid Generation on a Composite-Block Grid," AIAA Journal, vol. 28, pp. 470-477.

Leonard, B.P. (1979), "A Stable and Accurate Convective Modeling Procedure Based on Quadratic Upstream Interpolation," Computer Methods in Applied Mechanics and Engineering, vol. 19, pp. 59-98.

Leonard, B.P. (1988a), "Elliptic Systems: Finite-Difference Method IV," Handbook of Numerical Heat Transfer, Wiley, New York, pp. 347-378.

Leonard, B.P. (1988b), "Simple High-Accuracy Resolution Program for Convective Modelling of Discontinuities," International Journal for Numerical Methods in Fluids, vol. 8, pp. 1291-1318.

Leschziner, M.A. (1980), "Practical Evaluation of Trhee Finite Difference Schemes for the Computation of Steady-State Recirculating Flows," Computer Methods in Applied Mechanics and Engineering, vol. 23, pp. 293-312.

Malone, J.G. (1988), "Automated Mesh Decomposition and Concurrent Finite Element Analysis for Hypercube Multiprocessor Computers," Computer Methods in Applied Mechanics and Engineering, vol. 70, pp.27-58.

Moukalled, F.H. (1987), "Adaptive Grid Solution Procedure for Elliptic Flows," Ph.D. Thesis, Lousiana State University, Baton Rouge, LA.

Moukalled, F.H. and Acharya, S., (1991), "A Local Adaptive Grid Procedure for Incompressible Flows With Multigridding and Equidistribution Concepts," International Journal for Numerical Methods in Fluids, vol. 12.

Nietubicz, C.S. Heavey, K.R. and Steger, J.L. (1982), "Grid Generation Techniques for Projectile Configurations," in ARO Report 82-3 Proceedings of 1982 Army Numerical Analysis and Computer Conference, Vicksburg, Mississippi.

Oden, J.T. and Demkowicz, L. (1988), "Advances in Adaptive Improvements: A survey of Adaptive Finite Element Methods in Computational Mechanics," State of the Art Surveys in Computational Mechanics, eds. A.K. Noor and J.T. Oden, ASME, New York.

Oden, J.T., Rachowicz, W. and Kennon, S.R. (1991), "Numerical Analysis of Three-Dimensional Compressible Navier-Stokes Equations Using an Adaptive h-p Finite Element Method," AIAA paper 91-0119 presented at the 29th Aerospace Sciences Meeting, Reno, Nevada.

Patankar, S.V. (1980),Numerical Heat Transfer and Fluid Flow, Washington: Hemisphere Publishing Corporation.

Patel, M.K., Markatos, N.C., and Cross, M. (1985), "Methods of Reducing False-Diffusion Errors in Convection-Diffusion Problems," Applied Mathematical Modelling, vol. 9, pp. 301-306.

Patera, A. (1986), "Advances and Future Directions of Research on Spectral Methods," Computational Mechanics Advances and Trends, ed. A.K. Noor, ASME Publication AMD-Vol. 75, pp. 411-428.

Phillips, R.E. and Schmidt, F.W. (1984), "Multigrid Techniques for the Numerical Solution of the Diffusion Equation," Numerical Heat Transfer, vol. 7, pp. 251-268.

Phillips, R.E., Miller, T.F. and Schmidt, F.W. (1985), "A Multilevel-Multigrid Algorithm for Axisymmetric Recirculating Flows," Fifth Symposium on Turbulent Shear Flows, Cornell University, Ithaca, New York.

Prakash, C. and Patankar, S.V. (1981), "Combined Free and Forced Convection in Vertical Tubes with Radial Internal Fins," Journal of Heat Transfer, vol. 103, pp. 556-572.

Quinn, M.J. (1987), Designing Efficient Algorithms for Parallel Computers, McGraw-Hill, New York.

Rai, M.M. and Anderson, D.A. (1981), "Grid Evolution in Time Asymptotic Problems," Journal of Computational Physics, pp. 327-344.

Rai, M.M. (1988), "Applications of Domain Decomposition Methods to Turbomachinery Flows," presented at the Symposium on Advances and Applications in Computational Fluid Dynamics, Winter Annual Meeting of ASME.

Raithby, G.D. (1976), "A Critical Evaluation of Upstream Differencing Applied to Problems Involving Fluid Flow," Computer Methods in Applied Mechanics and Engineering, vol. 9, p. 75.

Rhie, C.M. (1986), "A Pressure Based Navier-Stokes Solver Using the Multigrid Method," AIAA Paper 86-0207.

Rogers, S.E. and Kwak, D. (1990), "Upwind Differencing Scheme for Time-Accurate Incompressible Navier-Stokes Equations," AIAA Journal, vol. 28, no. 2, pp. 253-262.

Saltzman, J. and Brackbill, J. (1982), "Applications and Generalizations of Variational Methods for Generating Adaptive Meshes," Numerical Grid Generation, ed. J.F. Thompson, North-Holland.

Schreiber, R. and Keller, H.B. (1983), "Driven Cavity Flows by Efficient Numerical Techniques," Journal of Computational Physics, vol. 49, pp. 310-333.

Settari, A. and Aziz, K. (1986), "A Generalization of the Additive Correction Methods for the Iterative Solution of Matrix Equations," SIAM Journal of Numerical Analysis, vol. 10, pp. 511-538.

Spalding, D.B. (1972), "A Novel Finite-Difference Formulation for Differential Expressions Involving Both First and Second Derrivatives," International Journal for Numerical Methods in Engineering, vol. 4, p. 551.

Thompson, J.F., Warsi, Z.U.A. and Martin, C.W. (1982), "Boundary-Fitted Coordinate Systems for Numerical Solution of Partial Differential Equations - A Review," Journal of Computational Physics, vol. 47, pp. 1-108.

Thompson, J.F. (1983), "A Survey of Grid Generation Techniques in Computational Fluid Dynamics," AIAA Paper 83-0447 presented at AIAA 21st Aerospace Sciences Meeting, Reno, Nevada.

Ushimary, K. (1982), "Development and Application of Adaptive Grids in Two-Dimensional Transonic Calculations," AIAA Paper 82-1016 presented at AIAA/ASME 3rd Joint Thermophysics, Fluids, Plasma and Heat Transfer Conference, St. Louis, Missouri.

Vanka, S.P. (1986), "Block Implicit Multigrid Solution of Navier-Stokes Equations in Primitive Variables," Journal of Computational Physics, vol. 65, pp. 138-158.

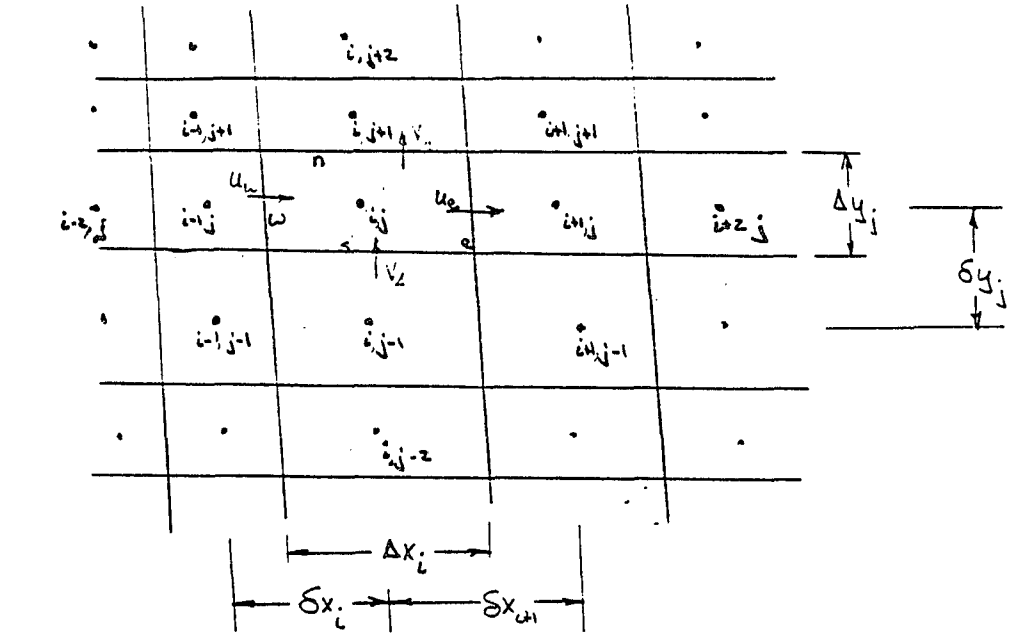
Venkatakrishnan, V. (19xx), "Preconditioned Conjugate Gradient Methods for the Compressible Navier-Stokes Equations,"

Wang, M. and Georgiadis, J.G. (1990), "Parallel Computation of Forced Convection on a Shared-Memory MIMD Computer," presented at AIAA/ASME Thermophysics and Heat Transfer Conference, Seattle, Washington.

APPENDIX

The following is a brief derrivation of Eqs. 2.14 and 2.15.

VARIABLE GRID FORMULATION/ TR



With Transverse Curvature terms (following Han, et al)

$$\phi_e = \frac{1}{2}\left(\phi_{ij} + \phi_{i+1j}\right) - \frac{1}{8}(\delta x_{i+1})^2 \, CURVN + \frac{1}{24}(\Delta y_j)^2 CURVT$$

$$u_e \geq 0, \quad CURVN = \frac{1}{\Delta x_i}\left[\frac{1}{\delta x_i}\left(\phi_{i-1j} - \phi_{ij}\right) - \frac{1}{\delta x_{i+1}}\left(\phi_{ij} - \phi_{i+1j}\right)\right]$$

$$u_e < 0, \quad CURVN = \frac{1}{\Delta x_{i+1}}\left[\frac{1}{\delta x_{i+2}}\left(\phi_{i+2} - \phi_{i+1j}\right) - \frac{1}{\delta x_{i+1}}\left(\phi_{i+1} - \phi_{ij}\right)\right]$$

$$u_e \geq 0, \quad CURVT = \frac{1}{\Delta y_j}\left[\frac{1}{\delta y_{j+1}}\left(\phi_{ij+1} - \phi_{ij}\right) - \frac{1}{\delta y_j}\left(\phi_{ij} - \phi_{ij-1}\right)\right]$$

$u_e < 0$ $\quad CURVT = \frac{1}{\Delta y_j}\left[\frac{1}{\delta y_{j+1}}(\phi_{i+1\,j+1} - \phi_{i+1\,j}) - \frac{1}{\delta y_j}(\phi_{i+1\,j} - \phi_{i+1\,j-1})\right]$

IF IPOSE = 1, INEGE = 0 for $u_e \geq 0$ and IPOSE = 0, INEGE = 1 for $u_e < 0$,

then,

$$\phi_e = \frac{IPOSE}{8}\left[4\phi_{ij} + 4\phi_{i+1\,j} + \frac{1}{\Delta x_i}(\delta x_{i+1})^2\left(\frac{\phi_{ij}}{\delta x_i} + \frac{\phi_{i+1\,j}}{\delta x_{i+1}}\right)\right]$$

$$- \frac{IPOSE}{8}\frac{(\delta x_{i+1})^2}{\Delta x_i}\left[\frac{\phi_{i-1\,j}}{\delta x_i} + \frac{\phi_{i+1\,j}}{\delta x_{i+1}}\right]$$

$$+ \frac{IPOSE}{24}\frac{(\Delta y_j)^2}{\Delta y_j}\left[\frac{\phi_{i\,j+1}}{\delta y_{j+1}} + \frac{\phi_{i\,j-1}}{\delta y_j} - \phi_{ij}\left(\frac{1}{\delta y_{j+1}} + \frac{1}{\delta y_j}\right)\right]$$

$$+ \frac{INEGE}{8}\left[4\phi_{ij} + 4\phi_{i+1\,j} - \frac{(\delta x_{i+1})^2}{\Delta x_{i+1}}\left(\frac{\phi_{ij}}{\delta x_{i+1}}\right)\right]$$

$$- \frac{INEGE}{8}\frac{(\delta x_{i+1})^2}{\Delta x_{i+1}}\left[\frac{\phi_{i+2\,j}}{\delta x_{i+2}} - \phi_{i+1\,j}\left(\frac{1}{\delta x_{i+2}} + \frac{1}{\delta x_{i+1}}\right)\right]$$

$$+ \frac{INEGE}{24}\frac{(\Delta y_j)^2}{\Delta y_j}\left[\frac{\phi_{i+1\,j+1}}{\delta y_{j+1}} + \frac{\phi_{i+1\,j-1}}{\delta y_j} - \phi_{i+1\,j}\left(\frac{1}{\delta y_{j+1}} + \frac{1}{\delta y_j}\right)\right]$$

$$\phi_w = \frac{1}{2}(\phi_{ij} + \phi_{i-1\,j}) - \frac{1}{8}(\delta x_i)^2 CURVN + \frac{1}{24}(\Delta y_j)^2 CURVT$$

For $u_w \geq 0$,

$$CURVN = \frac{1}{\Delta x_{i-1}}\left[\frac{1}{\delta x_{i-1}}(\phi_{i-2\,j} - \phi_{i-1\,j}) - \frac{1}{\delta x_i}(\phi_{i-1\,j} - \phi_{ij})\right]$$

$$CURVT = \frac{1}{\Delta y_j}\left[\frac{1}{\delta y_{j+1}}(\phi_{i-1\,j+1} - \phi_{i-1\,j}) - \frac{1}{\delta y_j}(\phi_{i-1\,j} - \phi_{i-1\,j-1})\right]$$

for $u_\omega \le 0$:

$$\omega RVN = \frac{1}{\Delta x_i}\left[\frac{1}{\delta x_i}(\phi_{i-1\,j} - \phi_{ij}) - \frac{1}{\delta x_{i+1}}(\phi_{ij} - \phi_{i+1\,j})\right]$$

$$\omega RVT = \frac{1}{\Delta y_j}\left[\frac{1}{\delta y_{j+1}}(\phi_{ij+1} - \phi_{ij}) - \frac{1}{\delta y_j}(\phi_{ij} - \phi_{ij-1})\right]$$

or

$$\phi_\omega = \frac{IPOSW}{8}\left[4\phi_{ij} + 4\phi_{i-1\,j} - \frac{\delta x_i}{\Delta x_{i-1}}\phi_{ij}\right]$$

$$- \frac{IPOSW}{8}\frac{(\delta x_i)^2}{\Delta x_{i-1}}\left[\frac{\phi_{i-2\,j}}{\delta x_{i-1}} - \phi_{i-1\,j}\left(\frac{1}{\delta x_{i-1}} + \frac{1}{\delta x_i}\right)\right]$$

$$+ \frac{IPOSW}{24}(\Delta y_j)\left[\frac{\phi_{i-1\,j+1}}{\delta y_{j+1}} + \frac{\phi_{i-1\,j-1}}{\delta y_j} - \phi_{i-1\,j}\left(\frac{1}{\delta y_{j+1}} - \frac{1}{\delta y_j}\right)\right]$$

$$+ \frac{INEGW}{8}\left[4\phi_{ij} + 4\phi_{i-1\,j} + \frac{(\delta x_i)^2}{\Delta x_i}\phi_{ij}\left(\frac{1}{\delta x_i} + \frac{1}{\delta x_{i+1}}\right)\right]$$

$$- \frac{INEGW}{8}\frac{(\delta x_i)^2}{\Delta x_i}\left[\frac{\phi_{i+1\,j}}{\delta x_i} + \frac{\phi_{i+1\,j}}{\delta x_{i+1}}\right]$$

$$+ \frac{INEGW}{24}(\Delta y_j)\left[\frac{\phi_{i\,j+1}}{\delta y_{j+1}} + \frac{\phi_{i\,j-1}}{\delta y_j} - \phi_{ij}\left(\frac{1}{\delta y_{j+1}} + \frac{1}{\delta y_j}\right)\right]$$

$$\phi_n = \frac{1}{2}(\phi_{ij} + \phi_{ij+1}) - \frac{1}{8}(\delta y_{j+1})^2 \, CURVN + \frac{1}{24}(\Delta x_i)^2 \, CURVT$$

for $V_n \geq 0$:

$$CURVN = \frac{1}{\Delta y_j}\left[\frac{1}{\delta y_j}(\phi_{ij-1} - \phi_{ij}) - \frac{1}{\delta y_{j+1}}(\phi_{ij} - \phi_{ij+1})\right]$$

$$CURVT = \frac{1}{\Delta x_i}\left[\frac{1}{\delta x_i}(\phi_{i-1j} - \phi_{ij}) - \frac{1}{\delta x_{i+1}}(\phi_{ij} - \phi_{i+1j})\right]$$

for $V_n < 0$:

$$CURVN = \frac{1}{\Delta y_{j+1}}\left[\frac{1}{\delta y_{j+2}}(\phi_{ij+2} - \phi_{ij+1}) - \frac{1}{\delta y_{j+1}}(\phi_{ij+1} - \phi_{ij})\right]$$

$$CURVT = \frac{1}{\Delta x_i}\left[\frac{1}{\delta x_i}(\phi_{i-1j+1} - \phi_{ij+1}) - \frac{1}{\delta x_{i+1}}(\phi_{ij+1} - \phi_{i+1j+1})\right]$$

or regrouping (as per Han, et al):

$$\phi_n = \frac{IPOSN}{8}\left[4\phi_{ij} + 4\phi_{ij+1} + \frac{(\delta y_{j+1})^2}{\Delta y_j}\phi_{ij}\left(\frac{1}{\delta y_j} + \frac{1}{\delta y_{j+1}}\right)\right]$$

$$- \frac{IPOSW}{8}\frac{(\delta y_{j+1})^2}{\Delta y_j}\left[\frac{\phi_{ij-1}}{\delta y_j} + \frac{\phi_{ij+1}}{\delta y_{j+1}}\right]$$

$$+ \frac{IPOSW}{24}\frac{(\Delta x_i)^2}{\Delta x_i}\left[\frac{\phi_{i-1j}}{\delta x_i} + \frac{\phi_{i+1j}}{\delta x_{i+1}} - \phi_{ij}\left(\frac{1}{\delta x_i} + \frac{1}{\delta x_{i+1}}\right)\right]$$

$$+ \frac{INEGN}{8}\left[4\phi_{ij} + 4\phi_{ij+1} - \frac{(\delta y_{j+1})^2}{\Delta y_{j+1}}\frac{\phi_{ij}}{\delta y_{j+1}} -\right.$$

$$-\frac{INEGN}{8}\frac{(\delta y_{j+1})^2}{\Delta y_{j+1}}\left[\frac{\phi_{i,j+2}}{\delta y_{j+2}} - \phi_{i,j+1}\left(\frac{1}{\delta y_{j+2}} + \frac{1}{\delta y_{j+1}}\right)\right]$$

$$+\frac{INEGN}{24}\frac{(\Delta x_i)^2}{\Delta x_i}\left[\frac{\phi_{i-1,j+1}}{\delta x_i} + \frac{\phi_{i+1,j+1}}{\delta x_{i+1}} - \phi_{i,j+1}\left(\frac{1}{\delta x_i} + \frac{1}{\delta x_{i+1}}\right)\right]$$

$$\phi_\Delta = \frac{1}{2}(\phi_{ij} + \phi_{i,j-1}) - \frac{1}{8}(\delta y_j)^2 \, CURVN + \frac{1}{24}(\Delta x_i)^2 \, CURVT$$

for $V_\Delta \geq 0$:

$$CURVN = \frac{1}{\Delta y_{j-1}}\left[\frac{1}{\delta y_{j-1}}(\phi_{i,j-2} - \phi_{i,j-1}) - \frac{1}{\delta y_j}(\phi_{i,j-1} - \phi_{ij})\right]$$

$$CURVT = \frac{1}{\Delta x_i}\left[\frac{1}{\delta x_i}(\phi_{i-1,j-1} - \phi_{i,j-1}) - \frac{1}{\delta y_{i+1}}(\phi_{ij} - \phi_{i+1,j-1})\right]$$

for $V_\Delta \leq 0$:

$$CURVN = \frac{1}{\Delta y_j}\left[\frac{1}{\delta y_j}(\phi_{i,j-1} - \phi_{ij}) - \frac{1}{\delta y_{j+1}}(\phi_{ij} - \phi_{i,j+1})\right]$$

$$CURVT = \frac{1}{\Delta x_i}\left[\frac{1}{\delta x_i}(\phi_{i-1,j} - \phi_{ij}) - \frac{1}{\delta x_{i+1}}(\phi_{ij} - \phi_{i+1,j})\right]$$

so,

$$\phi_\Delta = \frac{IPOSS}{8}\left[4\phi_{ij} + 4\phi_{ij-1} - \frac{(\delta y_j)}{\Delta y_{j-1}}\phi_{ij}\right]$$
$$- \frac{IPOSS}{8}\frac{(\delta y_j)^2}{\Delta y_{j-1}}\left[\frac{\phi_{i,i-2}}{\delta y_{j-1}} - \phi_{ij-1}\left(\frac{1}{\delta y_{j-1}} + \frac{1}{\delta y_j}\right)\right]$$
$$+ \frac{IPOSS}{24}(\Delta x_i)\left[\frac{\phi_{i-1,j-1}}{\delta x_i} + \frac{\phi_{i+1,j-1}}{\delta x_{i+1}} - \phi_{ij-1}\left(\frac{1}{\delta x_i} + \frac{1}{\delta x_{i+1}}\right)\right]$$
$$+ \frac{INEGS}{8}\left[4\phi_{ij} + 4\phi_{ij-1} + \frac{(\delta y_j)^2}{\Delta y_j}\phi_{ij}\left(\frac{1}{\delta y_j} + \frac{1}{\delta y_{j+1}}\right)\right]$$
$$- \frac{INEGS}{8}\frac{(\delta y_j)^2}{\Delta y_j}\left[\frac{\phi_{ij-1}}{\delta y_j} + \frac{\phi_{ij+1}}{\delta y_{j+1}}\right]$$
$$+ \frac{INEGS}{24}(\Delta x_i)\left[\frac{\phi_{i-1,j}}{\delta x_i} + \frac{\phi_{i+1,j}}{\delta x_{i+1}} - \phi_{ij}\left(\frac{1}{\delta x_i} + \frac{1}{\delta x_{i+1}}\right)\right]$$

Substituting back into the discretized equation

$$F_e\phi_e - F_w\phi_w + F_n\phi_n - F_\Delta\phi_\Delta = D_e(\phi_{i+1,j} - \phi_{ij}) - D_w(\phi_{ij} - \phi_{i-1,j})$$
$$+ D_n(\phi_{ij+1} - \phi_{ij}) - D_\Delta(\phi_{ij} - \phi_{ij-1}) + (S_c + S_p\phi_{ij})\Delta x_i \Delta y_j$$

and
regrouping like terms,

$$a_{ij}\phi_{ij} = b_{ij}\phi_{i-1,j} + c_{ij}\phi_{i+1,j} + d_{ij}\phi_{ij-1} + e_{ij}\phi_{ij+1} + f_{ij}$$

where

$$b_{ij} = F_w\left(\frac{IPOSW}{8}\right)\left[4 + \frac{(\delta x_i)^2}{\Delta x_{i-1}}\left(\frac{1}{\delta x_{i-1}} + \frac{1}{\delta x_i}\right)\right]$$

$$+ \frac{F_w}{8} INEGW\left[4 - \frac{(\delta x_i)^2}{\Delta x_i}\frac{1}{\delta x_i}\right] + D_w$$

$$c_{ij} = -\frac{F_e}{8} IPOSE(4) - \frac{F_e}{8} INEGE(4) + D_e = -\frac{F_e}{2}(IPOSE + INEGE) + D_e$$

$$c_{ij} = -0.5 F_e + D_e$$

$$d_{ij} = \frac{F_s}{8} IPOSS\left[4 + \frac{(\delta y_j)^2}{\Delta y_{j-1}}\left(\frac{1}{\delta y_{j-1}} + \frac{1}{\delta y_j}\right)\right]$$

$$+ \frac{F_s}{8} INEGS\left[4 - \frac{(\delta y_j)^2}{\Delta y_j}\left(\frac{1}{\delta y_j}\right)\right] + D_s$$

$$e_{ij} = -\frac{F_n}{8} IPOSN(4) - \frac{F_n}{8} INEGN(4) + D_n = -\frac{F_n}{2}(IPOSN + INEGN) + D_n$$

$$e_{ij} = -0.5 F_n + D_n$$

$$a_{ij} = -S_p \Delta x_i \Delta y_j + D_e + D_n + D_s - 0.5(F_w + F_s)$$
$$+ \frac{F_e}{8}\left[IPOSE\left(4 + \frac{\delta x_{i+1}}{\Delta x_i}\left(\frac{\delta x_{i+1}}{\delta x_i} + 1\right)\right) + INEGE\left(4 - \frac{\delta x_{i+1}}{\Delta x_{i+1}}\right)\right]$$
$$+ \frac{F_n}{8}\left[IPOSN\left(4 + \frac{\delta y_{j+1}}{\Delta y_j}\left(\frac{\delta y_{j+1}}{\delta y_j} + 1\right)\right) + INEGN\left(4 - \frac{\delta y_{j+1}}{\Delta y_{j+1}}\right)\right]$$

$$f_{ij} = S_c \Delta x_i \Delta y_j + F_e \left(\frac{IPOSE}{8}\right)\frac{(\delta x_{i+1})^2}{\Delta x_i}\left[\frac{\phi_{i-1,j}}{\delta x_i} + \frac{\phi_{i+1,j}}{\delta x_{i+1}}\right]$$
$$+ f_e \left(\frac{INEGE}{8}\right)\frac{(\delta x_{i+1})^2}{\Delta x_{i+1}}\left[\frac{\phi_{i+2,j}}{\delta x_{i+2}} - \phi_{i+1,j}\left(\frac{1}{\delta x_{i+2}} + \frac{1}{\delta x_{i+1}}\right)\right]$$
$$- F_w \left(\frac{IPOSW}{8}\right)\frac{(\delta x_i)^2}{\Delta x_{i-1}}\left[\frac{\phi_{i-2,j}}{\delta x_{i-1}} + \frac{\phi_{ij}}{\delta x_i}\right]$$
$$- F_w \left(\frac{INEGW}{8}\right)\frac{(\delta x_i)^2}{\Delta x_i}\left[\frac{\phi_{i+1,j}}{\delta x_{i+1}} - \phi_{ij}\left(\frac{1}{\delta x_i} + \frac{1}{\delta x_{i+1}}\right)\right]$$
$$+ F_n \left(\frac{IPOSN}{8}\right)\frac{(\delta y_{j+1})^2}{\Delta y_j}\left[\frac{\phi_{i,j-1}}{\delta y_j} + \frac{\phi_{i,j+1}}{\delta y_{j+1}}\right.$$

$$+ F_n \left(\frac{IPOSN}{8}\right) \frac{(\delta y_{j+1})^2}{\Delta y_j} \left[\frac{\phi_{i,j-1}}{\delta y_j} + \frac{\phi_{i,j+1}}{\delta y_{j+1}}\right]$$

$$+ F_n \frac{INEGN}{8} \frac{(\delta y_{j+1})^2}{\Delta y_{j-1}} \left[\frac{\phi_{i,j+2}}{\delta y_{j+2}} - \phi_{i,j+1}\left(\frac{1}{\delta y_{j+1}} + \frac{1}{\delta y_{j-2}}\right)\right]$$

$$- F_\Delta \frac{IPOSS}{8} \frac{(\delta y_j)^2}{\Delta y_{j-1}} \left[\frac{\phi_{i,j-2}}{\delta y_{j-1}} + \frac{\phi_{ij}}{\delta y_j}\right]$$

$$- F_\Delta \frac{INEGS}{8} \frac{(\delta y_j)}{\Delta y_j} \left[\frac{\phi_{i,j+1}}{\delta y_{j+1}} - \phi_{ij}\left(\frac{1}{\delta y_j} + \frac{1}{\delta y_{j+1}}\right)\right]$$

$$- F_e \frac{IPOSE}{24} \Delta y_j \left[\frac{\phi_{i,j+1}}{\delta y_{j+1}} + \frac{\phi_{i,j-1}}{\delta y_j} - \phi_{ij}\left(\frac{1}{\delta y_j} + \frac{1}{\delta y_{j+1}}\right)\right]$$

$$- F_e \frac{INEGE}{24} \Delta y_j \left[\frac{\phi_{i+1,j+1}}{\delta y_{j+1}} + \frac{\phi_{i+1,j-1}}{\delta y_j} - \phi_{i+1,j}\left(\frac{1}{\delta y_j} + \frac{1}{\delta y_{j+1}}\right)\right]$$

$$+ F_w \frac{IPOSW}{24} \Delta y_j \left[\frac{\phi_{i-1,j+1}}{\delta y_{j+1}} + \frac{\phi_{i-1,j-1}}{\delta y_j} - \phi_{i-1,j} \left(\frac{1}{\delta y_j} + \frac{1}{\delta y_{j+1}} \right) \right]$$

$$+ F_w \frac{INEGW}{24} \Delta y_j \left[\frac{\phi_{i,j+1}}{\delta y_{j+1}} + \frac{\phi_{i,j-1}}{\delta y_j} - \phi_{ij} \left(\frac{1}{\delta y_j} + \frac{1}{\delta y_{j+1}} \right) \right]$$

$$- F_n \frac{IPOSN}{24} \Delta x_i \left[\frac{\phi_{i+1,j}}{\delta x_{i+1}} + \frac{\phi_{i-1,j}}{\delta x_i} - \phi_{ij} \left(\frac{1}{\delta x_i} + \frac{1}{\delta x_{i+1}} \right) \right]$$

$$- F_n \frac{INEGN}{24} \Delta x_i \left[\frac{\phi_{i+1,j+1}}{\delta x_{i+1}} + \frac{\phi_{i-1,j+1}}{\delta x_i} - \phi_{i,j+1} \left(\frac{1}{\delta x_i} + \frac{1}{\delta x_{i+1}} \right) \right]$$

$$+ F_s \frac{IPOSS}{24} \Delta x_i \left[\frac{\phi_{i+1,j-1}}{\delta x_{i+1}} + \frac{\phi_{i-1,j-1}}{\delta x_i} - \phi_{i,j-1} \left(\frac{1}{\delta x_i} + \frac{1}{\delta x_{i+1}} \right) \right]$$

$$+ F_s \frac{INEGS}{24} \Delta x_i \left[\frac{\phi_{i+1,j}}{\delta x_{i+1}} + \frac{\phi_{i-1,j}}{\delta x_i} - \phi_{ij} \left(\frac{1}{\delta x_i} + \frac{1}{\delta x_{i+1}} \right) \right]$$

VITA

Therese E. Rhodes was born on August 27, 1959 in Louisville, Kentucky. She completed her high school education in Tuscaloosa, Alabama in May of 1977. She received her Bachelor of Science in Mechanical Engineering from the University of Alabama in May, 1981. During her entire undergraduate studies, she was supported by a scholarship from the College of Engineering. She returned to the University of Alabama to complete a Masters of Science in Engineering Mechanics in August, 1983. In August, 1986 she returned to graduate school at Lousiana State University to pursue a doctoral degree where she was awarded a four-year fellowship from the university.

DOCTORAL EXAMINATION AND DISSERTATION REPORT

Candidate: Therese E. Rhodes

Major Field: Mechanical Engineering

Title of Dissertation: An Adaptive Differencing Scheme for Elliptic Flows

Approved:

Major Professor and Chairman

Dean of the Graduate School

EXAMINING COMMITTEE:

Date of Examination:

June 6, 1991